Vrije Universiteit Brussel

Private Cloud with Raspberry Pi

Project Operating Systems and Security

Laurent Declercq

June 2016

Abstract

A trend towards cloud computing is going on for several years now. Along with this trend, several issues and concerns regarding privacy and security emerged. This report starts off by briefly explaining the current challenges of traditional computing, together with the corresponding cloud solutions that moreover explain this rising trend.

Mainly, the practice of ‘cloud computing’ is clearly defined and explained, together with a variety of user options. One particular option is looked at in-depth: establishing a private cloud using a Raspberry Pi. The reader is guided with a step-by-step tutorial for setting up the necessary configurations. This setup is then compared with other popular commercial cloud services.

Table of content

Table of figures

1. Introduction

1.1. Cloud Computing defined

The Cloud has become a greatly overloaded term over the last few years. It covers many services and technologies that have been around for some time and that are not necessarily offered together (Winans & Brown, 2009). Large cloud vendors offer data center hosting sometimes combined with web applications, interoperability with other applications and much more. Sometimes even the Internet as a whole is referred to as 'The Cloud'. To provide clarity, the cloud concept, shown in the picture below, is defined by NIST (National Institute of Standards and Technology) to create a framework that can be used to classify specific services (Mell & Grance, 2011). It splits the cloud into three service models, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

In **SaaS**, access to end user applications, such as Gmail, is provided over the Internet or intranet through a thin-client interface. The applications themselves actually run in a datacenter and not on the local terminal (Apprenda, 2016). The **PaaS** model is designed for software developers. It offers development tools for web and mobile applications over the Internet or intranet. Examples are Red Hat OpenShift and Google App Engine. Finally, **IaaS**, can be compared to home utilities such as electricity. Storage and processing power is delivered over the Internet or intranet. This allows for on-demand sharing of data, online backups and much more. However, it is the customer's responsibility to manage everything that is running on the server (Kabir, Islam, & Hossain, 2015).

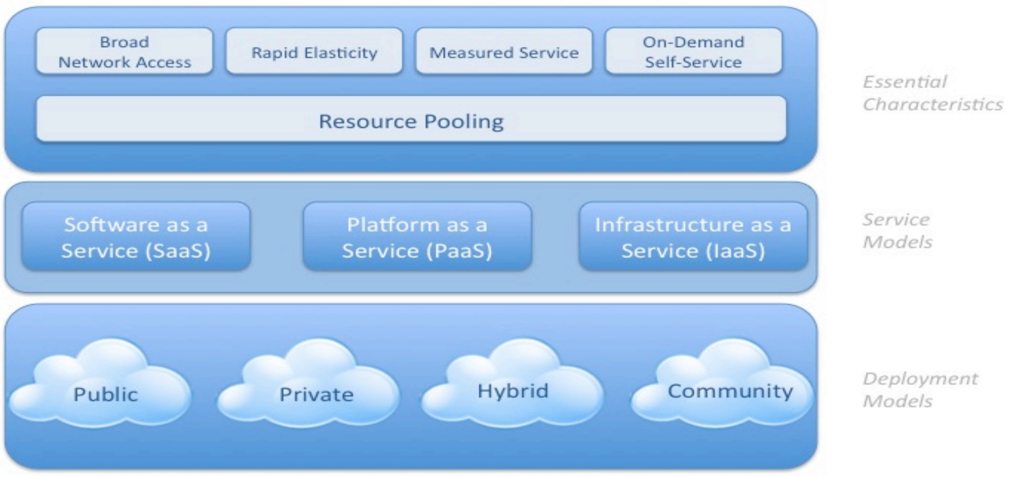


Figure 1 : Visual Model of NIST Definition of Cloud Computing

Another distinction must be made regarding the deployment models between public, private or hybrid clouds. In public clouds, the providers offer their physical resources to multiple organizations and/or consumers simultaneously. It can host both a collection of services and individual services. In private clouds, these resources are not shared. In this case, the service providers dedicate specific computing power, storage and other resources to a single client. We also talk about private clouds when a company/consumer has created its own cloud by investing in resources such as a datacenter. It can then be referred to as an on-premises cloud. Public clouds can be offered more cheaply since the providers can make use of economies of scale. However, compared to private clouds, the clients have less control over things such as resource allocation and other configurations. Solutions exist that combine the cost-effectiveness of public clouds with the security and configurability of private clouds. These are called hybrid clouds.

Characteristics that apply to all cloud service models and deployment models are on-demand self-service, resource pooling, broad network access rapid elasticity and measured service. According to the NIST definition of cloud computing, a service must have these essential characteristics to be labeled as a cloud solution. 'On-demand self-service' is the ability to hide and adapt the limitations of the system so that it appears to be infinite to the users. 'Resource pooling' means that, for multiple users, the resources are combined and assigned dynamically to improve efficiency. With 'broad network access', the resources are available over the Internet through multiple platforms. 'Rapid elasticity' is achieved when the resources can be scaled easily to cope with changing demand. Next to that, the characteristic of 'measured service' is one that can be seen on any cloud service. The user receives certain limited resources based on the subscription model. The usage of these resources is shown to the user.

1.2. Industrial Context

In the current fast-paced markets, it is of the essence that companies are flexible to adapt to new opportunities and threats. Therefore, focusing on the core business is increasingly important. Services that merely exist to support the delivery of the core products and core services become subject to outsourcing. In many cases a specialized company can provide the same service more cost-effectively, qualitatively and efficiently. When this is the case, outsourcing is done when it doesn't pose a threat to the company's intellectual property. For almost all companies, IT infrastructure makes up the backbone of their entire operation. Sometimes it includes an in-house built cloud. Factories and other facilities across borders rely on secure connections for day-to-day communication and operation. Establishing and maintaining such infrastructure requires significant investments and in-house knowledge. Although important, in most cases it is not the core business of a company. Cloud vendors today are able to offer the same, and better, infrastructure in a much more cost-effective way.

1.2.1. Current issues for corporate IT

Data Centers require large investments and are thus fairly fixed assets. With a fixed capacity, it cannot cope with the unbalanced workload that arises from the complexity of current global markets. Also, for startups and SME's, it is difficult, if not impossible, to finance such an investment. As business applications are costly to create/buy and integrate within the company, many of them are outdated and lack the ability to communicate with other application platforms outside the company. This, in turn, can put a strain on overall business performance.

1.2.2. Cloud Solutions

The cloud gives answers to these issues (Kabir et al., 2015). Cloud vendors, thanks to their large scale, are able to efficiently handle spikes in demand. They offer scalability and agility and can afford to keep updating their systems to state-of-the-art technology. Next to that, geographical distribution offers many opportunities in network efficiency but also energy efficiency and cost-savings. Demand can easily be shifted to parts of the world where it is night or to a location where it can currently draw solar power or any other cost-effective source of energy (green computing).

Application portfolio management is also improved thanks to virtualisation, by which the partners can be presented with a standardized platform that, in the short-term draws upon all their current applications, in order for the transition to newer technologies to go as smooth as possible.

Overall, cloud computing offers resource efficiency and a modernized and up-to-date IT application portfolio in the short term. Moreover, it offers long-term business advantages such as agility. An example would be an acquisition of another company. Now, it is a painstaking process for the IT department to integrate different systems. These problems are easily overcome by the use of a distributed platform. Financially, the business is also more agile because high capital investments are exchanged with much lower operational costs. This leaves financial room for other business needs.

1.3. Security concerns

Many of the basic features that make cloud services so appealing, also make up the foundation of some major security concerns (Mazhar Ali, Khan, & Vasilakos, 2015). Only some concerns are described here. Cloud solutions offer scalability by pooling resources. Many users are tied to the same resource set. This leads to the risk of data exposure to other users. Next to that, in this shared environment, the security of the entire service is only as good as its weakest link. An intruder might compromise the whole service from this weak link. These shared resources together with scalability inevitably lead to one client using resources that were once allocated to another client. Using data recovery techniques, sensitive data from previous users can easily be exposed (Balduzzi, Zaddach, Balzarotti, & Loureiro, 2012).

Next to that, the different cloud service models are interdependent. IaaS can be seen as the basis upon which the Paas is built. Similarly, the applications of the Saas are developed using the PaaS tools. Intruders can take advantage of these dependencies by attacking one model and compromising the other service models that are left exposed. A more obvious risk source are the service providers themselves. Employees with bad intentions might be granted sufficient access to pose a potential threat.

From this, it is clear that there are still many security, privacy and regulatory threats that need to be addressed by the cloud vendors. For large corporations, many of these concerns can be tackled in service level agreements (SLA) with the vendor (M Ali et al., 2015). This way they can already make use of many cloud technologies before planning a full transition to cloud applications and services. Still, legal issues become somewhat harder to tackle when the data is migrated to a datacenter located in a country with different laws.

1.4. Consumer Context

Large corporations have the power to secure many potential risks of cloud computing by means of tailored contracts negotiated with cloud vendors. This involves ownership of data, IP protection and law enforcement in case of security breaches and much more. In contrast, most individual users don’t have this negotiation power and are left with many concerns. These include the reliability of the service, data security, privacy, increased dependency upon a third party, fairness and transparency of the unnegotiable terms and conditions and so forth (Cunningham, 2013).

Most cloud service providers operate under a freemium model. Limited services are offered for free and the user receives the option to pay for subscription fees depending on the services needed, such as extra storage. The user however, becomes increasingly dependent upon the services that are linked to one another, often including cloud storage, web mail and calendar applications. Therefore, consumers are increasingly led into paying more for cloud service.

Another problem with free cloud services comes from the need for a revenue stream. The service provider can achieve this by collecting and selling user data. This in itself is not necessarily a problem. The problem arises when the transparency of these transactions are questionable. Mostly, users are not aware which data is shared with whom and might find themselves more exposed than anticipated.

1.4.1. Private Cloud Solutions (consumers)

Setting up your own cloud can thus be very beneficial for a couple of reasons. Cost and privacy reasons have been discussed above. Three important service platforms iCloud, Google Drive and Dropbox all handle similar pricing schemes that include free basic storage and monthly fees for extended storage. Google Drive takes the lead here with 15 GB free storage, which is 3 times more than iCloud's base storage and almost 8 times more than what Dropbox offers for free, 2 GB. For 1 TB of storage, you will pay €9,99 per month on all of these platforms. Given that subscribing for the 1TB plan for 1 year would cost you more than buying a 1 TB hard drive, for many people, the choice is quickly made.

The amount of features is platform dependent but many include a sharing option, auto-sync to local folders and calendar synchronization. Luckily, there are many options available for creating a private cloud with similar features. Solutions range from more expensive ready-to-use NAS devices (Network Attached Storage, cf. below), to very cheap or free software that can be installed on basically any system to achieve similar goals. All these solutions are based on the same core elements, a storage device, processing power, Internet connectivity and software.

A potential downside can be speed. Whereas file transfer speed within the local network is faster than downloading files over the Internet, outside the boundaries of your network this is usually not the case. For every download from your private cloud, the transfer rate is limited by the upload speed established by your ISP. With the most common ISP's in Belgium, Telenet and Proximus, consumer upload speeds are limited, especially compared to download speeds. Telenet offers upload speeds of 3Mbps, 6Mbps or 12 Mbps, depending on the service plan, whereas Proximus' highest upload speed lies at 6 Mbps. Downloading your files from a cloud service is (after uploading it at the usual upload speed) only limited by your download speed offered by the ISP, which can easily be 100 Mbps.
NAS

A NAS or network-attached storage device is a simple computer, specifically aimed at serving files to a single or multiple users. In many cases, it is used to also provide data redundancy through the use of a RAID array. Here, depending on the RAID level, the data is automatically distributed across the drives. The files are shared using multiple network file sharing protocols such as AFP (Mac OSX), NFS (UNIX) or SMB/CIFS (Windows). Compared to existing cloud services, a NAS is much more sophisticated on certain domains. It has similar storing and sharing capabilities but next to that, it can act as a multimedia server, a print server or a small database server. However, it lacks other features that many cloud services offer such as deduplication and block-level compression.

NAS boxes are often sold with a bundle of applications and are accessible through a 'simple' web interface. QNAP, Synology and Western Digital are well established names in the NAS market. However, with these boxes customizability is limited, prices tend to be high and energy consumption is high as well. That is why, you might be better of building a NAS from scratch. Our aim here is to build a system similar to current cloud offerings, for personal use and occasional sharing. Requirements for this system are fairly low and so a Raspberry Pi will do the job.

2. Project Hardware and Software

The following section explains the components needed for a private cloud. Next to that, it specifies the options and the choices that were made. For this project, a Raspberry Pi 1 Model B is used, running Raspbian as operating system. There is an NGINX server running on it, as well as the file-hosting service OwnCloud and the AFP file-sharing service Netatalk. To make it easier to connect to the server permanently, we installed no-ip client software. To provide some level of data redundancy, a daily backup process is scheduled at night using rsync and crontab.

2.1. Raspberry Pi

The Raspberry Pi is a low cost, low energy mini-computer that is aimed at teachers and students. With it, they are introduced to programming and computers in general. It can be used as any other computer to browse the web, read e-mails etc., but in most cases it will be used for specific projects and for prototyping. When correctly set-up, it can be used as a NAS box, very much like the of-the-shelve-boxes.

For this project we will be using the original Raspberry Pi, the Pi 1 model B. The newer versions are similar but offer improvements in performance and memory size. The original version is currently available for +/- €20 and has the following specifications (Makershed, 2015):

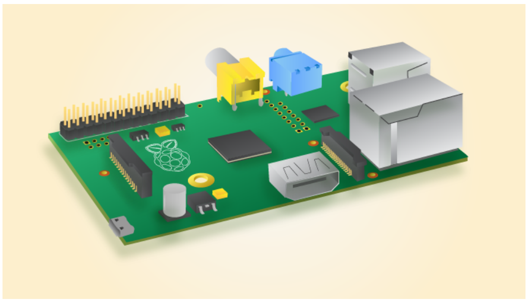


Figure 2 : Raspberry Pi 1 model B

Chip: Broadcom BCM2835
Processor: ARMv6 single core
Processor speed: 700 MHz
Power consumption: 600 mA @ 5V
Memory: 512 MB SDRAM @ 400 MHz
2 USB 2.0 ports
1 SD card slot
26 GPIO pins
Ethernet: 10/100mb
GPU: Dual Core VideoCore IV Multimedia Co-Processor

2.2. Operating System: Raspbian

There are operating systems out there that are specifically tailored for building home storage devices. They are slimmed-down versions of general-purpose operating systems, focusing on file sharing protocols and features (Whitson, 2014).

Amahi offers a straightforward user interface that includes one-click installers of popular applications such as ownCloud and Plex. It is based on Linux so it is possible to install native Linux apps as well.

FreeNAS is a sophisticated home server OS, based on FreeBSD. It supports RAID and many other things but requires at least 1GB RAM per TB of storage that you install, which the Raspberry Pi 1 does not have. An alternative would be NAS4Free, which is an old, less sophisticated version of FreeNAS. It is more suited for low-powered devices.

We will use Raspbian, based on Debian, because it is optimized for the Raspberry Pi. It offers the flexibility to install other native Linux packages that are, in turn, also optimized for the Pi. Next to that, it is fairly well documented.

2.3. File hosting service: ownCloud

ownCloud is a free and open-source suite of file hosting services with which your data is synchronised among all your connected device. It offers the same features as Dropbox and many more such as contact synchronisation. Since it is open-source, applications can easily be made to expand the platform. Examples include in-browser video streaming and a photo gallery. Alternatives for ownCloud are Pydio, Sparkleshare and Seafile. Pydio has a nicer GUI than ownCloud, better performance for large data but fewer documentation at the moment. Sparkleshare is more focused towards documents, as it offers build in version control as well as a revision history. Seafile is still very much under development and has less support for file sharing protocols.

ownCloud is basically an application running on the operating system. To be available online, a web server must be installed as well.

2.4. Web Server: NGINX

For accessing files that are stored on your Pi over the Internet via an Internet browser, we will need a web server. It stores, processes and delivers web pages to clients. Basically, a client, being a web browser, asks for a web page or a specific resource by sending an HTTP GET request to the server. The server, in turn, responds by sending the resource via HTTP. To enable uploading of files, the web server can also receive content from its clients.

There are two HTTP servers that are so popular that they make up half of the traffic on the Internet. These are Apache and NGINX (pronounced Engine X). Both are open source and can provide the same basic server solutions. However, they both have their unique features that make them better suited in specific situations (Ellingwood, 2015). Apache is older and is chosen for its flexibility, power and extendibility. It can deliver static and dynamic content internally but has difficulties coping with a large amount of concurrent connections. NGINX, on the other hand, released in 2004, focuses on scalability in order to meet the requirements of thousands of simultaneous connections. With its light-weight architecture it makes very efficient use of its resources. NGINX only handles static content internally. Dynamic content delivery (such as PHP requests), however, is a task that is passed on to other modules. This means that NGINX needs to communicate with the processor via some protocol such as memcache. For the project, this protocol needs to be installed as well. This has the advantage that for static content, less overhead is incurred. The two HTTP servers also differ on many other aspects such as their module system, their mapping of requests to resources and their configuration systems.

2.5. Scripting language: PHP5

Pages and documents can be delivered statically or dynamically. With static web pages, the user simply views a page but cannot alter the information on that page. There is no interaction. The web page is an HTML document that exists at the server-side and which is transferred to the browser when a request comes in. However, this means that the web page does not necessarily show the correct state of the documents that you want to view. This is crucial for good file-sharing services. Next to that, no changes will be seen on the page after deleting or uploading files.

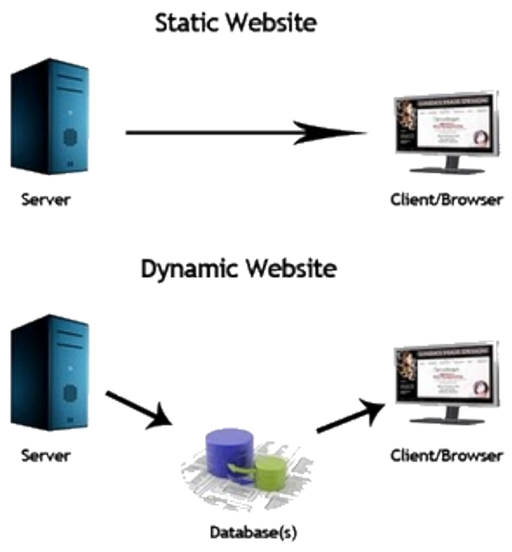


Figure 3 : Static vs Dynamic Website schema

Dynamic web pages provide a solution for this. They create a live user experience. They can be provided in two ways, either by client-side scripting or server-side scripting.

A server-side dynamic web page is built on-demand by the web server using a scripting language such as PHP, ASP, Perl, etc, to retrieve the necessary information and combine it into an HTML document. Most web servers support scripting languages to achieve this goal. PHP5 will be used to create our cloud environment.

With a client-side dynamic web page, the browser itself creates the interaction within the page. Scripting languages such as JavaScript are used to process the web page as it loads and to then reload to change the variable content.

2.6. Dynamic DNS service: no-ip

Every Internet-connected household receives an IP address from the Internet service provider (ISP). These IP addresses are mostly dynamic, meaning that the service provider redistributes the IP addresses after a period of time, using the Dynamic Host Configuration Protocol (DHCP). This is done partly to minimize the amount of used IP addresses, given that not all clients will be online at the same time. However, this poses a problem when trying to access your private cloud from outside the local network. All Internet domain names are linked to an IP address, using a table called the Domain Name System (DNS). However, when the IP address of the server, running that domain name changes, the table is not adjusted automatically. After such a change has occurred, you won’t be able to connect anymore.

A Dynamic DNS (DDNS or DynDNS) service links and updates the most current IP address to connect to a hostname in the DNS. This works by installing a client program on the device that is running the server and setting it up using the same login-information as used on the website of the DDNS provider. The the DDNS client sends the correct IP address to the service, which adapts its binding to the domain name, ready to provide it to anyone that enters the domain name.

2.7. File sharing protocol: AFP (Netatalk)

SMB (“Server Message Block”), has been the main Windows File Sharing protocol since the 90s. The protocol, although developed by IBM, is proprietary to Microsoft. However, it was revers engineered to create the Samba protocol. Samba allows unix-based systems to share files in a Windows environment and was implemented by Apple in OS X 10.2. Apple itself however, already used another protocol for file-sharing among macs (unix), called AFP (“Apple[Talk] Filing Protocol”). AFP outperforms SMB in numerous ways including read/write performance and sleep suspend support (JPY, 2015).

With Windows Vista, Microsoft has released a new version of SMB, SMB2, which enhances speed, reliability and security. So much even that Apple introduced its own implementation of the SMB2 protocol, named SMBX (Dilger, 2013). It has been pushing SMBX to be its default file-sharing protocol and would only fall back onto AFP for devices that don't support SMB(X). This way, Windows compatibility is strongly improved. More recently SMB3 was released, with many features not present in AFP such as encryption and SMB multi-channel (Ryan, 2016). Apple will likely decrease support for AFP in favor of SMB3 in the upcoming years. For now, however, AFP is still a great choice in a mac-only environment thanks to its performance, simplicity and documentation. Netatalk has been developed as an open source AFP3.3 implementation. This package will be installed on the Raspberry Pi to fulfill our file-sharing needs.

Other file-sharing options include the older FTP, NFS and WebDAV. FTP "File Transfer Protocol" is an old protocol that is known for its weak security (Steiner, 2010). NFS (Network File System) also lacks decent security. WebDAV (Web Distributed Authoring and Versioning) is an extension of HTTP with certain file-sharing capabilities. It is mostly used in one of its extensions such as CalDAV, developed for remote calendar access; and CardDAV, which is used to share an address book on a server. Development for WebDAV stopped in 2007.

2.8. Data redundancy: Rsync

Rsync (Remote sync) is used to copy and/or synchronise directories and files, both locally and remotely (Shrivastava, 2013). It minimizes the required effort by transferring only the differences between two sets of files. It finds where the differences are located within a file and transfers only those data blocks. The blocks that need to be transferred, are first compressed to consume less bandwidth. By default, rsync compares files and directories on both ends based on the 'last modified date' and the size of the file. This will miss differences that don't appear in these two attributes. The probability of these type of differences occurring is very slow in a normal user environment. In environments where this occurs often, or where no mistakes are tolerated, the option --checksum offers a thorough checksum comparison. This requires more time to process.

3. Methodology

This section is written as a tutorial in order to provide clarity in case this project needs to be reproduced. It is based on multiple tutorials with similar goals in mind. First, ownCloud is installed (Gus, 2015a; Koff1979, 2012; Mayank, 2016; Young, 2015) and made available over the Internet (Gus, 2015b). Next, direct file sharing is enabled (Damontimm, 2010; Thijxx, 2012). Furthermore, data redundancy is provided with rsync.

3.1. Hardware used

- Raspberry Pi 1 Model B
- SD card (4 GB Minimum)
- Ethernet cable
- 2 USB thumb drives
- Mac to SSH onto the Raspberry Pi

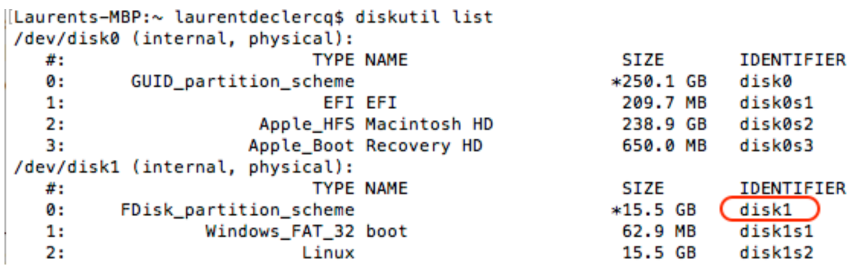
3.2. Installing ownCloud

3.2.1. Installing Raspbian (OS) on the Raspberry Pi

After downloading the Raspbian zip file from https://www.raspberrypi.org/downloads/, it should be decompressed to a .img file. After that, it can be written to the SD card. For this, a script called Pi Filler can be used or the process below can be followed. The steps below are a slight adaptation of a tutorial on the official Raspberry Pi website.

Open the terminal.
Run the following command to identify the disk number of your SD card:

```
diskutil list
```



```
Laurents-MBP:~ laurentdeclercq$ diskutil list
/dev/disk0 (internal, physical):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      GUID_partition_scheme                        *250.1 GB   disk0
   1:                        EFI EFI                     209.7 MB   disk0s1
   2:                  Apple_HFS Macintosh HD            238.9 GB   disk0s2
   3:                 Apple_Boot Recovery HD             650.0 MB   disk0s3
/dev/disk1 (internal, physical):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:     FDisk_partition_scheme                        *15.5 GB    disk1
   1:             Windows_FAT_32 boot                    62.9 MB    disk1s1
   2:                      Linux                         15.5 GB    disk1s2
```

Figure 4 : diskutil list command

In this case, the drive is numbered disk1 (not disk1s1, because that is the partition).
To copy data to the SD card, it first has to be unmounted. Replace disk1 below with the number found with the previous command:

```
diskutil unmountDisk /dev/disk1
```

Copy the downloaded .img file to your SD card. My download was located in the Downloads folder. Replace the path below to fit your location, and disk1 with your disk number:

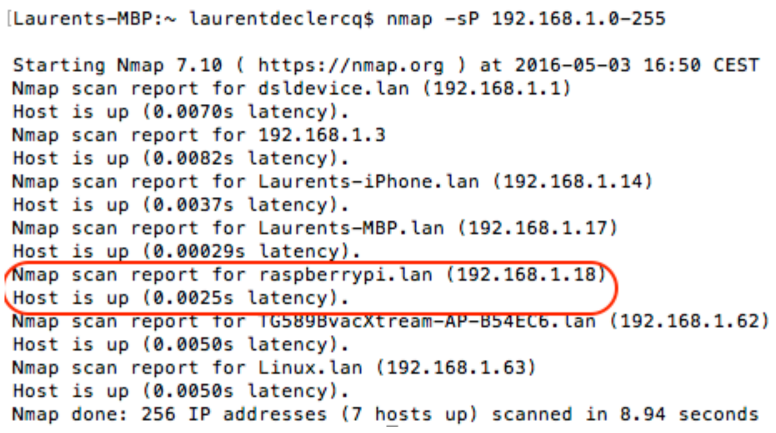
```
sudo dd bs=1m if=Downloads/2016-03-18-raspbian-jessie.img of=/dev/rdisk1
```

Copying the data can take a few minutes up to a few hours. To follow the progress, a SIGINFO signal can be sent by pressing Ctrl+T.
Put the SD card in the Raspberry Pi, connect it to power and connect it to your router via the Ethernet cable.

3.2.2. Find IP of Raspberry PI and Login over SSH

SSH uses the IP address of the device you want to connect to. We can use a tool called NMap to find the Pi's IP address. Instead of using the IP address, the name 'raspberrypi' can also be used to log in. Still, it is important to know the IP address of the Raspberry Pi since this will be needed in future steps. If you don't know the IP address of your router by heart, it will also be listed. This will be needed later on as well. Normally it should have 192.168.1.1 or 192.168.0.1 as the default IP address.

```
nmap -sP 192.168.1.0-255
```



```
Laurents-MBP:~ laurentdeclercq$ nmap -sP 192.168.1.0-255

Starting Nmap 7.10 ( https://nmap.org ) at 2016-05-03 16:50 CEST
Nmap scan report for dsldevice.lan (192.168.1.1)
Host is up (0.0070s latency).
Nmap scan report for 192.168.1.3
Host is up (0.0082s latency).
Nmap scan report for Laurents-iPhone.lan (192.168.1.14)
Host is up (0.0037s latency).
Nmap scan report for Laurents-MBP.lan (192.168.1.17)
Host is up (0.00029s latency).
Nmap scan report for raspberrypi.lan (192.168.1.18)
Host is up (0.0025s latency).
Nmap scan report for TG589BvacXtream-AP-B54EC6.lan (192.168.1.62)
Host is up (0.0050s latency).
Nmap scan report for Linux.lan (192.168.1.63)
Host is up (0.0050s latency).
Nmap done: 256 IP addresses (7 hosts up) scanned in 8.94 seconds
```

Figure 5 : nMap command

Connect to it with SSH, default username is *pi* and the default password is *raspberry.* In the line below, use the IP address of your raspberry:

`ssh pi@192.168.1.18` or simply `ssh pi@raspberrypi`

3.2.3. Setting up the Raspberry Pi Server and preparing for ownCloud

First update the Pi and all its packages to avoid errors down the line.

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

We will adjust the default configurations to better suit our needs. The settings can be found in the Raspi Config Tool.

```
sudo raspi-config
```

Change Locale to en_US.UTF8 in internationalisation options -> change local.
Change memory split to 16m in Advanced options -> Memory split.
Change overclock to medium.

Add a www-data user to the www-data group. This is a sort of passive user that cannot be logged on to but that is used for the backend file and directory sharing with the ownCloud service. This provides security in the sense that no person could guess the password because none has been set:

```
sudo usermod -a -G www-data www-data
```

Install the necessary packages, needed for ownCloud and general server operation:

```
sudo apt-get install nginx openssl ssl-cert php5-cli php5-sqlite php5-gd php5-common php5-
cgi sqlite3 php-pear php-apc curl libapr1 libtool curl libcurl4-openssl-dev php-xml-parser
php5 php5-dev php5-gd php5-fpm memcached php5-memcache varnish
```

Create an SSL certificate by entering the following command and filling in the requested information regarding location, organization and email.

```
sudo openssl req $@ -new -x509 -days 730 -nodes -out /etc/nginx/cert.pem -keyout
/etc/nginx/cert.key
```

Adjust the privileges to the two created cert files:

```
sudo chmod 600 /etc/nginx/cert.pem
sudo chmod 600 /etc/nginx/cert.key
```

Clear the server config file. We will copy and paste our own version in it.

```
sudo sh -c "echo '' > /etc/nginx/sites-available/default"
```

Open the configuration file of our NGINX web server with the text editor so we can make changes to it. We will copy and paste the configuration below so that NGINX works correctly with ownCloud.

```
sudo nano /etc/nginx/sites-available/default
```

Copy and paste the following and replace the IP on the line of server_name, with the IP of your Pi. In this case, the Raspberry Pi is located at 192.168.1.18. After that, save and exit.

```
upstream php-handler {
  server 127.0.0.1:9000;
  #server unix:/var/run/php5-fpm.sock;
}
server {
  listen 80;
  server_name 192.168.1.18;
  return 301 https://$server_name$request_uri;  # enforce https
}

server {
  listen 443 ssl;
  server_name 192.168.1.18;
  ssl_certificate /etc/nginx/cert.pem;
  ssl_certificate_key /etc/nginx/cert.key;
  # Path to the root of your installation
  root /var/www/owncloud;
  client_max_body_size 1000M; # set max upload size
  fastcgi_buffers 64 4K;
  rewrite ^/caldav(.*)$ /remote.php/caldav$1 redirect;
  rewrite ^/carddav(.*)$ /remote.php/carddav$1 redirect;
  rewrite ^/webdav(.*)$ /remote.php/webdav$1 redirect;
  index index.php;
  error_page 403 /core/templates/403.php;
  error_page 404 /core/templates/404.php;
  location = /robots.txt {
     allow all;
     log_not_found off;
     access_log off;
  }
  location ~ ^/(?:\.htaccess|data|config|db_structure\.xml|README) {
     deny all;
  }
  location / {
     # The following 2 rules are only needed with webfinger
     rewrite ^/.well-known/host-meta /public.php?service=host-meta last;
     rewrite ^/.well-known/host-meta.json /public.php?service=host-meta-json last;
     rewrite ^/.well-known/carddav /remote.php/carddav/ redirect;
     rewrite ^/.well-known/caldav /remote.php/caldav/ redirect;
     rewrite ^(/core/doc/[^\/]+/)$ $1/index.html;
     try_files $uri $uri/ index.php;
  }
  location ~ \.php(?:$|/) {
     fastcgi_split_path_info ^(.+\.php)(/.+)$;
     include fastcgi_params;
     fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
     fastcgi_param PATH_INFO $fastcgi_path_info;
     fastcgi_param HTTPS on;
     fastcgi_pass php-handler;
}
# Optional: set long EXPIRES header on static assets
location ~* \.(?:jpg|jpeg|gif|bmp|ico|png|css|js|swf)$ {
     expires 30d;
     # Optional: Don't log access to assets
     access_log off;
}
}
```

Next, the PHP configuration file needs to be modified so that it will allow reasonable file sizes. Here, it is changed to 2 GB:

```
sudo nano /etc/php5/fpm/php.ini
```

In this file we want to find and update the following lines. (Ctrl + w allows you to search):

```
upload_max_filesize = 2000M
post_max_size = 2000M
```

Once done, save and exit.

Now, the PHP www.conf file needs to listen to the localhost via port 9000 instead of the current path.

```
sudo nano /etc/php5/fpm/pool.d/www.conf
```

Change *listen = /var/run/php-fpm.sock* to *listen = 127.0.0.1:9000*
Save and Exit.

We will also edit the dphys-swapfile.

```
sudo nano /etc/dphys-swapfile
```

Change *CONF_SWAPSIZE=100* to *CONF_SWAPSIZE=512*
Save and Exit

Restart the Raspberry Pi with the following command:

```
sudo reboot
```

3.2.4. Setting up ownCloud

Install ownCloud. To do this we will create a directory for it, download and unpack the software and move it to the right directory. We need to change the permissions of the www-data user to be able to access the ownCloud folder. After that, we perform some cleanup, that is, deleting the downloaded compressed file.

```
sudo mkdir -p /var/www/owncloud
sudo wget https://download.owncloud.org/community/owncloud-9.0.2.tar.bz2
sudo tar xvf owncloud-9.0.2.tar.bz2
sudo mv owncloud/ /var/www/
sudo chown -R www-data:www-data /var/www
rm -rf owncloud owncloud-9.0.2.tar.bz2
```

Adjust the .htaccess file to support the same file size that we set earlier, namely 2 GB.

```
cd /var/www/owncloud
sudo nano .htaccess
```

Update the following three parameters like this:

```
php_value_upload_max_filesize  2000M
php_value_post_max_size        2000M
php_value_memory_limit 2000M
```

Save and exit.

Do the same with the .user.ini file:

```
sudo nano .user.ini
```

Update the following three parameters like this:

upload_max_filesize=2000M
post_max_size=2000M
memory_limit=2000M

Save and exit.

3.2.5. Setting up an external drive

We want ownCloud to use the external hard drive or USB drive automatically to store all the data. Therefore, it is very useful to automatically mount the drive. Also, the path of that drive needs to be linked to ownCloud combined with the necessary permissions. This is done in the following instructions:

If the drive is NTFS formatted, the NTFS package needs to be installed first:

```
sudo apt-get install ntfs-3g
```

Make a directory that you can mount to, we call ours 'ownclouddrive'.

```
sudo mkdir /media/ownclouddrive
```

We will need information about the server user and the drive to update the fstab file. The fstab (or file systems table) file is a configuration file that lists all available disk partitions and the specifications on how to initialize each of them in order to fit in the general file system structure.
Retrieve gid (group identifier) of the www-data user

```
id -g www-data
```

Retrieve UUID of the hard drive. This is unique for each drive, so the system will recognize it even if it is plugged into a different USB port.

```
ls -l /dev/disk/by-uuid
```

We will need the blue letters and numbers of the last part. It should have /sda1 at the end.
Now, adjust the fstab file based on the information we just retrieved.

```
sudo nano /etc/fstab
```

Insert and adjust the following line at the bottom of the file, using the found guid and UUID

UUID=5302-16F0 /media/ownclouddrive auto nofail,gid=33,umask=0007,dmask=0007,noatime 0 0

The umask and dmask here are important since they set the permissions for different users. Here, it is set so that anyone in the www-data group has permissions to read, write and execute files and directories within the /media/ownclouddrive directory. In a further section, when we establish direct file sharing, a user is created inside the www-data group to take advantage of these permissions.

Reboot the Raspberry Pi, after that, the drive should be automatically mounted.

3.2.6. First use & setup of ownCloud

Open the browser and go to your Pi's IP address. For me, it is 192.168.1.18. If a certificate warning pops up, simply accept this. At the first login, you should see an initial setup screen that prompts you to create an admin account. Fill in the boxes but wait to continue!

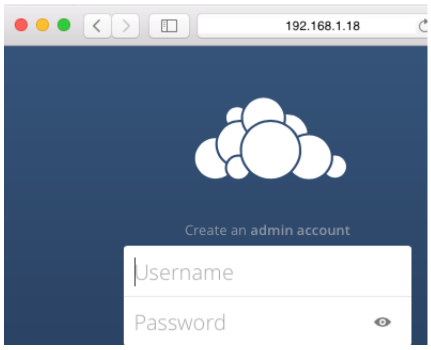


Figure 6 : ownCloud web interface

To set it up using your external drive, click on storage & database and enter the path to your external drive, being "/media/ownclouddrive". Beware, if you don't do this now, the process will be much more difficult afterwards. In case this warning did not help, the steps to change your data directory afterwards are included below.

Click Finish Setup.

3.2.7. Redirect server directory (in case forgotten during initial setup)

Even if you have set up your ownCloud already, you can change your data directory. This step can be skipped if it has already been done. The process goes as follows:
First stop the server:

```
sudo service nginx stop
```

Go to var/www/owncloud/config and change config.php

```
cd var/www/owncloud/config
sudo nano config.php
```

In order to point to the location where you want to have your data, change the datadirectory accordingly. Here, the data will be located at /media/ownclouddrive/usbdata, with usbdata a non-existing folder. The usbdata directory needs to be non-existing or the data will end up in a subdirectory of usbdata.

The files that reside in the directory that ownCloud first pointed towards, need to be moved to the new directory:

```
mv /var/www/owncloud/data /media/ownclouddrive/usbdata
```

Previously, the NGINX server was set-up to run as the www-data user. This same user needs to be granted the necessary ownership of the new directory.

```
chown –R www-data:www-data /media/ownclouddrive/usbdata
```

Start your webserver.

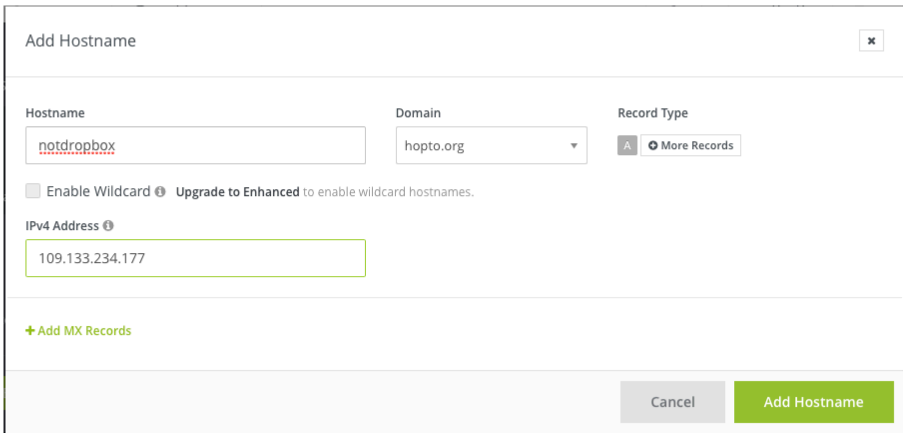
```
sudo service nginx start
```

3.2.8. Arrange external access

Up until now, the ownCloud platform has been available only locally. To provide a true Dropbox alternative, it is necessary to expand its reach over the Internet. This requires a few actions such as installing a DDNS service, reserving a local IP address for your raspberry PI, port forwarding and changing some configuration files that were used above.

DDNS configuration

As stated in the theoretical section of the report, noip will be used as a DDNS service (Auti, 2014). The DDNS service is needed when your ISP assigns you a public IP address that changes periodically. If this is not the case, this step can be skipped. First, you need to register a domain name at the website www.noip.com. We chose the domain name notdropbox.hopto.org. Notice that noip correctly binds this to the current external IP address. However, we need a Dynamic Update Client (DUC) that updates this binding periodically so that the domain name can keep redirecting the traffic to the changing IP address.


Add Hostname

Hostname: notdropbox

Domain: hopto.org

Record Type: A

Enable Wildcard — Upgrade to Enhanced to enable wildcard hostnames.

IPv4 Address: 109.133.234.177

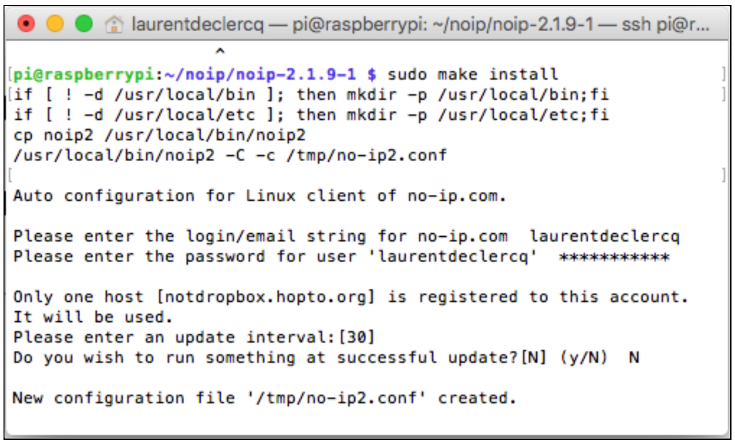
Figure 7 : no-ip DDNS configuration

NOIP client installation

In many routers, a DDNS client is already implemented. This can be checked by logging in to the router's settings page. In many cases, this can be reached under 192.168.1.1. or 192.168.0.1. Follow the guidelines on the DDNS settings page to synchronize it with your DDNS service provider. In case the router does not provide this service, a client needs to be installed so that the raspberry pi can provide this service itself.

First you need to create a directory for NOIP, go to that directory and download the NOIP software into that directory. After extracting, the downloaded file can be removed. Next, the client can be installed and started. After installing, it prompts you to fill in the login credentials used for the NOIP service. It then finds the domain name notdropbox.hopto.org. The other parameters will be left at default. With this, the client will send the IP address every 30 minutes. The following commands are required:

```
mkdir /home/pi/noip
cd /home/pi/noip
wget http://www.no-ip.com/client/linu/noip-duc-linux.tar.gz
tar vzxf noip-duc-linux.tar.gz
rm noip-duc-linux.tar.gz
cd noip-2.1.9-1
sudo make
sudo make install
```



laurentdeclercq — pi@raspberrypi: ~/noip/noip-2.1.9-1 — ssh pi@r...

```
pi@raspberrypi:~/noip/noip-2.1.9-1 $ sudo make install
if [ ! -d /usr/local/bin ]; then mkdir -p /usr/local/bin;fi
if [ ! -d /usr/local/etc ]; then mkdir -p /usr/local/etc;fi
cp noip2 /usr/local/bin/noip2
/usr/local/bin/noip2 -C -c /tmp/no-ip2.conf

Auto configuration for Linux client of no-ip.com.

Please enter the login/email string for no-ip.com  laurentdeclercq
Please enter the password for user 'laurentdeclercq'  ***********

Only one host [notdropbox.hopto.org] is registered to this account.
It will be used.
Please enter an update interval:[30]
Do you wish to run something at successful update?[N] (y/N)  N

New configuration file '/tmp/no-ip2.conf' created.
```

Figure 8 : noip client configuration

Start on reboot

To start it automatically, the rc.local file needs to be adapted:

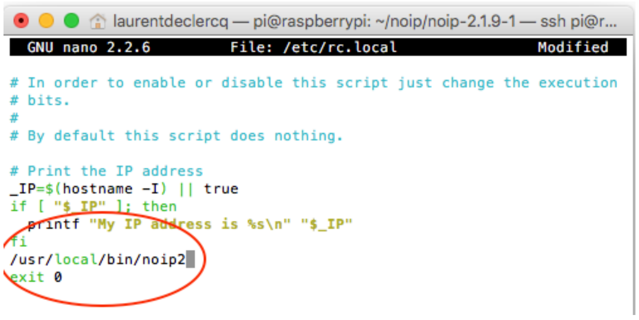
```
sudo nano /etc/rc.local
```

add the following: *\/usr/local/bin/noip2*

between the lines: *fi*

exit 0

so that it looks like this:



laurentdeclercq — pi@raspberrypi: ~/noip/noip-2.1.9-1 — ssh pi@r...

```
  GNU nano 2.2.6           File: /etc/rc.local                   Modified

# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi
/usr/local/bin/noip2
exit 0
```

Figure 9 : rc.local modification

NGINX reconfiguration

The NGINX configuration file should be changed as follows:

```
sudo nano /etc/nginx/sites-available/default
```

The server_name values now point to the local IP address of the Raspberry Pi. This needs to be changed into your external IP address, or your domain name:

Previously: *192.168.1.18*
Now: *notdropbox.hopto.org*

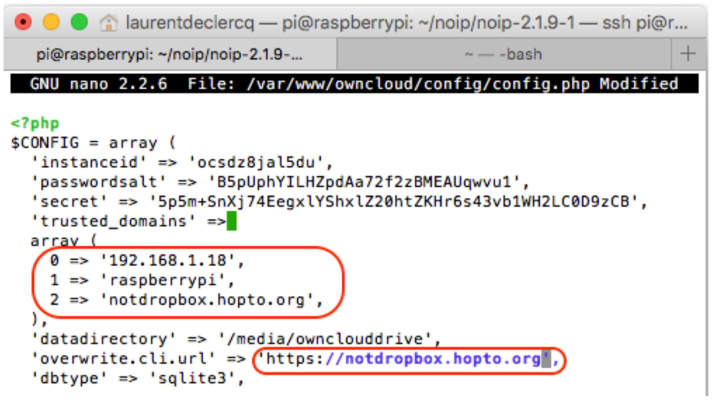
Also, some ISPs might block the use of port 80 (http) or port 443 (https) for personal use. For me, that was the case. It can save you a lot of trouble changing the listening port for https to something else like port 444, in the same configuration file. Check the availability of ports first to avoid conflicts. We will only use the secure https protocol. That is why port 80 can be left unchanged.

ownCloud reconfiguration

Now, ownCloud still needs to add the new domain to its trusted list. These domains are listed in the ownCloud config file:

```
sudo nano /var/www/owncloud/config/config.php
```

Simply add a line below the trusted domains with your external IP (either a domain name or an IP address, depending on whether or not your ISP assigns IP's dynamically). Also prevent ownCloud from overwriting it. The config file should look like this:


laurentdeclercq — pi@raspberrypi: ~/noip/noip-2.1.9-1 — ssh pi@r...
pi@raspberrypi: ~/noip/noip-2.1.9-...
~ — -bash

```
  GNU nano 2.2.6  File: /var/www/owncloud/config/config.php Modified

<?php
$CONFIG = array (
  'instanceid' => 'ocsdz8jal5du',
  'passwordsalt' => 'B5pUphYILHZpdAa72f2zBMEAUqwvu1',
  'secret' => '5p5m+SnXj74EegxlYShxlZ20htZKHr6s43vb1WH2LC0D9zCB',
  'trusted_domains' =>
  array (
    0 => '192.168.1.18',
    1 => 'raspberrypi',
    2 => 'notdropbox.hopto.org',
  ),
  'datadirectory' => '/media/ownclouddrive',
  'overwrite.cli.url' => 'https://notdropbox.hopto.org',
  'dbtype' => 'sqlite3',
```

Figure 10 : ownCloud config modification

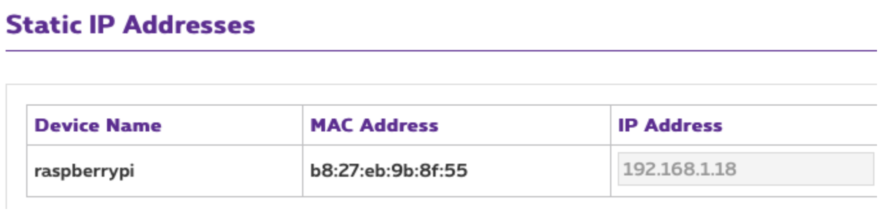
Restart the NGINX service:

```
sudo service nginx restart
```

Router reconfiguration

<u>DHCP reservation</u>

The router functions as a DHCP server, dynamically assigning IP addresses to the devices on your local network. Every time a client boots up, a DHCP lease negotiation occurs. The Raspberry Pi should be reachable at all times. NOIP makes sure our home network is reachable. However, the IP address of the raspberry Pi might still change due to a reboot. This needs to be avoided. The router can be set to manually assign IP addresses to specific clients. To do this, in your web browser, go to the IP address of your router. Login and go to DHCP. Depending on the manufacturer, the interface looks different but should have similar options. Add the Raspberry Pi to the list of static IP addresses.



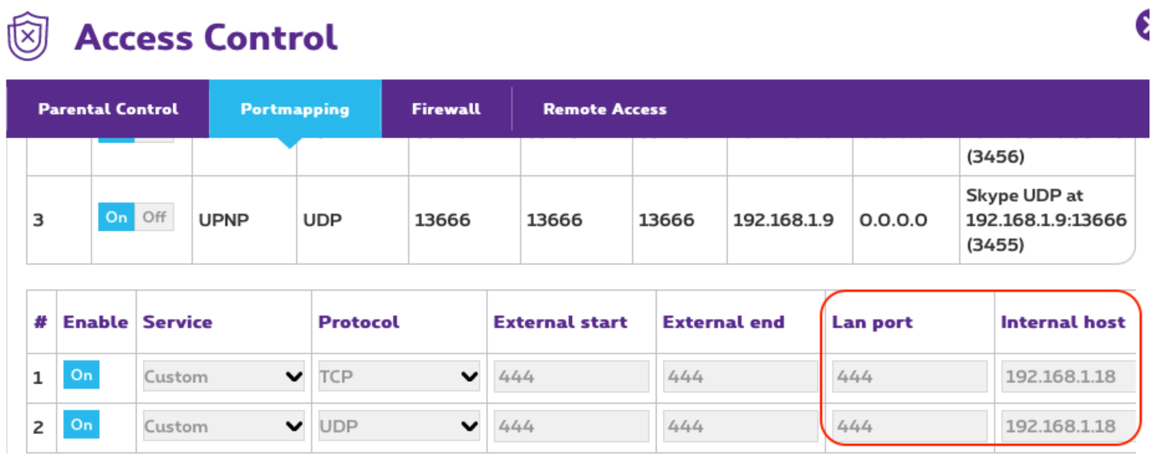
Static IP Addresses

Device Name	MAC Address	IP Address
raspberrypi	b8:27:eb:9b:8f:55	192.168.1.18

Figure 11 : DHCP reservation

<u>Portmapping</u>

Portmapping or port forwarding makes sure that the right applications can be reached. Again, the router settings can be modified so that port 444 (the https port, originally port 443) directs the traffic to the NGINX web server, running on the raspberry pi. The consequences of only adding this rule is that, when connecting to the ownCloud server externally you will need to make sure you use https. Simple http will lead to an invalid request in the browser.



									(3456)
3	On Off	UPNP	UDP	13666	13666	13666	192.168.1.9	0.0.0.0	Skype UDP at 192.168.1.9:13666 (3455)

#	Enable	Service	Protocol	External start	External end	Lan port	Internal host
1	On	Custom	TCP	444	444	444	192.168.1.18
2	On	Custom	UDP	444	444	444	192.168.1.18

Figure 12 : Portmapping router configuration

3.3. AFP file sharing

File sharing allows the raspberry Pi to share directories and files with specific users and groups. This makes it possible to work on any document as if it was located on your hard drive. This is very similar to the offerings of Dropbox and Google Drive. Our environment is a pure mac environment so there is no need for the SMB protocol. Also, as shown above, AFP is very fast and easy to set up, which is important for a good file-sharing service.

In addition to an AFP protocol, an implementation of Zeroconf (zero-configuration networking) will provide service discovery, address assignment and hostname resolution. This will basically advertise the shared drive on the local network. With this, the Raspberry Pi will show up as a network drive in the file explorer (Finder). Apple's proprietary implementation is called Bonjour, but an open source packet exists to imitate its behavior, called Avahi (Siddiqui, Zeadally, Kacem, & Fowler, 2012).

3.3.1. Create extra user

For our ownCloud project, the www-data user was created inside the www-data group. This is not a real user in the sense that is has no password set. This is because of security reasons. The only way to read and write files is via the ownCloud interface. However, in order to perform file sharing, we need to circumvent this safety regulation, without weakening its security. This can be done by creating a new user inside the www-data group (Rafacz, 2012). In the previous steps, the external hard drive is mounted automatically with read, write and execute permissions for all users in the www-data group. This new user will thus take advantage of these permissions. A new user is added as follows:

```
useradd –g www-data newuser
```

3.3.2. Installing Netatalk

Install the Netatalk package by entering the following command:

```
sudo apt-get install netatalk
```

Speed up the AFP server by turning off redundant support for mac OS 9:

```
sudo nano /etc/default/netatalk
```

change the file so that the options look like below:

```
ATALKD_RUN=no
PAPD_RUN=no
CNID_METAD_RUN=yes
AFPD_RUN=yes
TIMELORD_RUN=no
A2BOOT_RUN=no
```

Point Netatalk to the directory you wish to share. In our case, we are sharing the same folder that is visible on ownCloud to create one single shared directory on both interfaces (web and file explorer).

```
sudo nano /etc/netatalk/AppleVolumes.default
```

Add the following to the bottom, but changing the username. Add an extra line for every other directory that needs to be shared.

```
/media/ownclouddrive/laurentdeclercq/files "ownCloudDrive"
```

Options can be added on the line that's starts with *DEFAULT*. Adding *rw* allows for Read/Write access. The line should look somewhat like this:

:DEFAULT: options:upriv,usedots,rw

```
sudo /etc/init.d/netatalk restart
```

Although the file sharing is now working; the Raspberry Pi does not show up as a network device. It can be connected to by entering the IP-address in the 'Connect To Server' menu in the Finder. However, it would be much simpler to have it show up as a shared drive in the side bar. That is why we need a zeroconf implementation.

3.3.3. Avahi

Avahi is used as zeroconf implementation. It can be installed together with mDNS to better imitate Apple's proprietary Bonjour:

```
sudo apt-get install avahi-daemon libnss-mdns
```

```
sudo nano /etc/nsswitch.conf
```

append 'mdns' to the end of one line so that it looks like this:

hosts : files mdns4_minimal [NOTFOUND=return] dns mdns4 mdns

The AFP service is advertised by creating the following file with the content below:

```
sudo nano /etc/avahi/services/afpd.service
```

```
<?xml version="1.0" standalone='no'?><!--*-nxml-*-->
<!DOCTYPE service-group SYSTEM "avahi-service.dtd">
<service-group>
        <name replace-wildcards="yes">%h</name>
        <service>
                <type>_afpovertcp._tcp</type>
                <port>548</port>
        </service>
        <service>
                <type>_device-info._tcp</type>
                <port>0</port>
                <txt-record>model=Xserve</txt-record>
        </service>
```

</service-group>

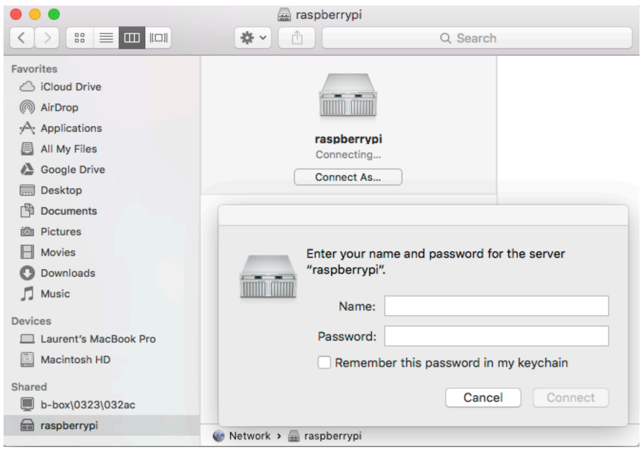
Save and exit.

The text above shows that AFP is advertised using port 548. Also the icon to be shown in the Finder is set at Xserve, an appropriate icon for a server. Other options are Macmini, iMac, MacPro, Xserve, MacBook, MacBookPro or MacBookAir.

Restart Avahi.

```
sudo /etc/init.d/avahi-daemon restart
```

Now the shared folder on the Raspberry Pi should appear as a network drive on any Mac in the network. You can log on with the extra user that was created in the www-data group. This way, you will have the right permissions to read and write in all subdirectories.

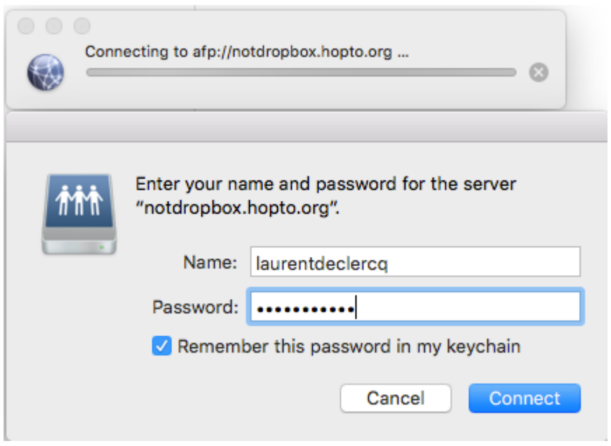


Enter your name and password for the server "raspberrypi".

Figure 13 : Finder view with shared Raspberry Pi drive

3.3.4. External access with AFP

As with the ownCloud platform, you can expand this file-sharing to access it from outside the network. This can be done easily now since the DDNS service and DHCP reservation have already been set-up. AFP uses port 548 so the router needs to be reconfigured to forward this port to the Raspberry Pi. For this, follow the steps in section 3.2.8 using port 548. After this, the network drive can be accessed by going to the Finder. In the GO menu, press ‘Connect to Server’ and enter afp://notdropbox.hopto.org, replacing it with your domain name. This will prompt you for the login credentials as seen in the figure below. Once entered, the drive will appear in the file explorer as if it were your local hard drive. For convenience, this drive can be set as a favorite.


Connecting to afp://notdropbox.hopto.org ...

Enter your name and password for the server "notdropbox.hopto.org".

Name: laurentdeclercq

Password: •••••••••••

☑ Remember this password in my keychain

Figure 14 : External access via AFP

3.4. Provide Data Redundancy

To provide data safety, having one copy of your file on one drive is not enough. Dropbox and other services use multiple copies to guarantee that, when certain drives fail, all the data will still be available. Thus to guarantee data safety and uptime on our private cloud, a true RAID level 5 or 6, or other backup mechanisms would be needed. However, for simplicity reasons and due to the lack of resources, basic data redundancy is achieved with a synchronization process that occurs overnight. Achieving data redundancy like this is done as follows (Fizpatrick, 2013):

First, an extra NTFS-formatted drive is needed.

Make a directory where the extra drive is going to be mounted to. Here it is called BACKUPDRIVE:

```
sudo mkdir /media/BACKUPDRIVE
```

Look for the UUID of the backup drive. The first hard drive show /sda1 at the end. The backup drive can be found in the line with /sdb1.

```
ls -l /dev/disk/by-uuid
```

Change the fstab file so that the extra drive is automatically mounted on reboot:

```
sudo nano /etc/fstab
```

Copy the line that was used for automatically mounting the owncloudDrive, but modify it to point to the new drive. It should look as follows:

```
UUID=5302-16F0 /media/ownclouddrive auto
nofail,gid=33,umask=0007,dmask=0007,noatime 0 0
UUID=67E3-17ED /media/BACKUPDRIVE auto
nofail,gid=33,umask=0007,dmask=0007,noatime 0 0
```

Save, exit and reboot.
Now, the rsync package can be installed.

```
sudo apt-get install rsync
```

Crontab allows to schedule specific tasks. In our case, it will only be used to schedule a backup process. It is best to schedule backups during the night so that it does not occupy bandwidth and processor time when you need them. To do this, open contrab and adjust the text document by adding the following line at the bottom, replacing it with your username and the correct name for the extra hard drive:

```
crontab –e
```

```
0 3 * * * rsync -rltzuv --delete /media/ownclouddrive/laurentdeclercq/files   /media/BACKUPDRIVE
```

Save and exit.

This tells contrab to start copying files from the ownclouddrive to the backupdrive at 3AM (0 3). It is scheduled to do this every day (* * * are wildcards, referring to year, month and day). The options –rltzuv refer to copying recursively, compressing the files before data transfer, etc. With the option --delete, it will delete anything on the backupdrive that cannot be found on the ownclouddrive.

To check that this statement works, and to lower the burden of the initial process overnight, the line above can be processed as a separate command:

```
rsync -rltzuv --delete /media/ownclouddrive/laurentdeclercq/files /media/BACKUPDRIVE
```

4. Results and Conclusion

4.1. Raspberry Pi setup vs to commercial cloud

Having followed all the steps provided above, has led to a product that can serve as a true alternative to commercial cloud services. The Raspberry offers a convenient way to access and share your files at home and on the go. With the ownCloud platform, the administrator has a wide set of tools to create user groups and appropriate permissions, link different calendars and contacts, offer live-collaboration, include chat, and many more. This way, ownCloud outperforms competitors such as Google Drive, that don't provide these administrator tools to adapt the service to your needs. As explained, the disadvantage of having the files reside on your network, is that the speed at which you can access them externally is limited to your upload speed. Externally, commercial cloud services clearly outperform the Raspberry Pi.

With the AFP file-sharing protocol, the Raspberry Pi offers the possibility to access your files anywhere right within your file explorer as if they were located on your hard drive. At home, this means that you can work at LAN speeds. These are, under normal circumstances, higher than download or upload speeds. Also, unlike Dropbox, the files are not stored on your local drive, leaving storage capacity for other data. Although less safe as Dropbox, the daily backup provides a simple way to avoid losing data in case one of the drives crashes. Added to this is the security argument of controlling your data in a private environment. No privacy issues can come up and as owner of your data, you know exactly where your data is and who might have access to it.

The use of the low-cost, low-powered Raspberry Pi means that interaction with the system can take quite some time. The latest version of the Raspberry Pi, the Pi 3, provides a much faster 1.2GHz 64-bit quad-core ARMv8 CPU, doubles the RAM memory to 1GB, adds connectivity features such as BLE and doubles the USB ports (4). This model is much more suited for the computationally expensive cloud tasks such as video streaming and photo editing.

4.2. Comparison to NIST definition for cloud computing

In order to be identified as a cloud solution, the Rasperry Pi configuration should answer certain criteria as stated in section 1.1. The comply with the NIST definition, the following characteristics need to be fulfilled: on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. The Raspberry fulfils limited **on-demand self-service** as it requires only adaptations from the administrator to add extra storage capacity when the current configuration seems to become insufficient. This means that, this requirement is only fulfilled in case of multiple users. Next, the **resource pooling** characteristic is not fulfilled. The system is not able to dynamically reassign its resources in a multi-tenant model. When multiple users access the system simultaneously, the system will treat these request similarly. However, **measured service** is a feature that ownCloud provides. The administrator can assign different storage capacity limits to different users and groups. The user is able to see how much has been used so far. The Raspberry Pi set-up provides limited **rapid elasticity** in the sense that it is relatively easy to change the storage capacity but the processing power is more fixed. When an expandable computer would have been used to set up the service, this would have been more easy to do. Finally, the system is available over the Internet via two different interfaces. This fulfills the **broad network access** requirement.

In practice, the Raspberry Pi with ownCloud and AFP file-sharing can be seen as a true Cloud computing service that is worth the effort of setting up.

5. Limitations

5.1. Automatic synchronisation with local folder

Here, we assume that the user wants to use a separate folder for files that need to be made available online. The user modifies the files and folders directly on the ownClouddrive itself. It would however, be equally possible to synchronise any local folder with the ownClouddrive using rsync in a similar fashion as how it is used to provide data redundancy. This way, the user keeps a local copy of the file, which is synchronized with the server in a background process. This has the advantage of independence of a reliable Internet connection. On the other hand, this requires local storage capacity.

5.2. Protocols

Here, AFP has been chosen because the Raspberry Pi acts in a mac-only environment. However, when multiple users want to interact with the system, it is more than likely that windows devices will be involved. In that case, SMB should be used for a reliable service. Another problem with AFP is that it works slowly when connecting over a slow or unreliable connection. Here again, it would be wise to use another protocol such as WebDAV.

6. References

Ali, M., Dhamotharan, R., Khan, E., Khan, S. U., Vasilakos, A. V, Li, K., & Zomaya, A. Y. (2015). SeDaSC: Secure Data Sharing in Clouds. *IEEE Systems Journal*. http://doi.org/10.1109/JSYST.2014.2379646

Ali, M., Khan, S. U., & Vasilakos, A. V. (2015). Security in cloud computing: Opportunities and challenges. *Information Sciences*, *305*, 357–383. http://doi.org/10.1016/j.ins.2015.01.025

Apprenda. (2016). IaaS, PaaS, SaaS (Explained and Compared). Retrieved February 1, 2016, from https://apprenda.com/library/paas/iaas-paas-saas-explained-compared/

Auti, D. (2014). How to Setup NOIP Client on Raspberry Pi. Retrieved April 1, 2016, from http://www.awesomeweirdness.com/projects-diy/raspberrypi/setup-noip-client-raspberry-pi/

Balduzzi, M., Zaddach, J., Balzarotti, D., & Loureiro, S. (2012). A Security Analysis of Amazon ' s Elastic Compute Cloud Service – Long Version. *Security*, 1427–1434. http://doi.org/10.1145/2245276.2232005

Cunningham, A. (2013). Caveat Consumer? – Consumer Protection and Cloud Computing – Part 2. *SSRN Electronic Journal*, 1–41. http://doi.org/10.2139/ssrn.2212051

Damontimm. (2010). How to set up AFP filesharing on Ubuntu. Retrieved October 2, 2016, from https://missingreadme.wordpress.com/2010/05/08/how-to-set-up-afp-filesharing-on-ubuntu/

Dilger, D. E. (2013). Apple shifts from AFP file sharing to SMB2 in OS X 10.9 Mavericks. Retrieved April 10, 2016, from http://appleinsider.com/articles/13/06/11/apple-shifts-from-afp-file-sharing-to-smb2-in-os-x-109-mavericks

Ellingwood, J. (2015). Apache vs Nginx: Practical Considerations.

Fizpatrick, J. (2013). How to Turn a Raspberry Pi into a Low-Power Network Storage Device. Retrieved June 8, 2016, from http://www.howtogeek.com/139433/how-to-turn-a-raspberry-pi-into-a-low-power-network-storage-device/

Gus. (2015a). Raspberry Pi OwnCloud: Your Own Personal Cloud Storage. Retrieved March 7, 2016, from https://pimylifeup.com/raspberry-pi-owncloud/

Gus. (2015b). Raspberry Pi Port Forwarding & Dynamic DNS. Retrieved March 16, 2016, from https://pimylifeup.com/raspberry-pi-port-forwarding/

JPY. (2015). AFP versus SMB/SAMBA. Retrieved May 20, 2016, from http://news.jpy.com/kbase/support-articles/2015/9/8/afp-versus-smbsamba

Kabir, M. H., Islam, S., & Hossain, S. (2015). A Detail Overview of Cloud Computing with its Opportunities and Obstacles in Developing Countries, *4*(4), 52–63.

Koff1979. (2012). Raspberry Pi Owncloud (dropbox clone). Retrieved March 10, 2016, from http://www.instructables.com/id/Raspberry-Pi-Owncloud-dropbox-clone/?ALLSTEPS

Makershed. (2015). Raspberry Pi Comparison Chart. Retrieved March 1, 2016, from http://www.makershed.com/pages/raspberry-pi-comparison-chart

Mayank, S. (2016). How to set up a Raspberry Pi-powered cloud service. Retrieved April 1, 2016, from http://www.techradar.com/how-to/computing/how-to-set-up-a-raspberry-pi-powered-cloud-service-1316017

Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. *National Institute of Standards and Technology, Information Technology Laboratory*, *145*, 7.

http://doi.org/10.1136/emj.2010.096966

Rafacz, R. (2012). Adding Linux Users and Groups. Retrieved May 13, 2016, from https://www.pluralsight.com/blog/tutorials/linux-add-user-command

Ryan, H. (2016). OS X Yosemite (Server 4.03) and SMB3. Retrieved April 10, 2016, from https://www.murage.ca/os-x-yosemite-server-4-03-smb3/

Shrivastava, T. (2013). Rsync (Remote Sync): 10 Practical Examples of Rsync Command in Linux. Retrieved June 9, 2016, from http://www.tecmint.com/rsync-local-remote-file-synchronization-commands/

Siddiqui, F., Zeadally, S., Kacem, T., & Fowler, S. (2012). Zero configuration networking: Implementation, performance, and security. *Computers and Electrical Engineering*, *38*(5), 1129–1145. http://doi.org/10.1016/j.compeleceng.2012.02.011

Steiner, P. (2010). Why FTP is no longer up to the job. *Network Security*, *2010*(5), 7–9. http://doi.org/10.1016/S1353-4858(10)70055-6

Thijxx. (2012). File sharing with OSX. Retrieved June 10, 2016, from https://www.raspberrypi.org/forums/viewtopic.php?f=36&t=26826

Whitson, G. (2014). What Operating System Should I Use for My DIY Home Server? Retrieved March 1, 2016, from http://lifehacker.com/what-operating-system-should-i-use-for-my-diy-home-serv-1671385076

Winans, T. B., & Brown, J. S. (2009). Cloud Computing: A Collection of Working Papers. *Http://Www.Johnseelybrown.Com*, 1–39. http://doi.org/10.1017/CBO9781107415324.004

Young, D. (2015). Create your own cloud server on the Raspberry Pi. Retrieved March 7, 2016, from https://www.element14.com/community/community/raspberry-pi/raspberrypi_projects/blog/2015/05/05/owncloud-v8-server-on-raspberry-pi-2-create-your-own-dropbox