NSS-Cloud Report

0 TOC

TBC Introduction?

This report discusses the inevitable heat death of the universe that is looming *almost* behind the corner.

1 System Goals

This section introduces primary goals of the system and serves as a basis for all decisions regarding the practical implementation of the system. The goals can generally be divided into two categories, qualitative goals and quantitative goals, though some may also fall into both. Additionally, we describe some potential use cases as keeping them in mind will allow us to design our system from the ground up to be used effectively in practice.

1.1 Qualitative Goals

Privacy is one of the core drivers and goals of this project. As more and more data, services, and even infrastructure (e.g. DNS resolution) are being centralized by the top players in the cloud industry, there is an increasing need for alternate options for those who need a guarantee of privacy and data confidentiality (resources TBC - confidential dns, confidential computing, GDPR). This need could be to some extent satisfied with personal cloud deployment on local network or self-hosted cloud.

Security goes hand-in-hand with privacy, as there cannot be any true privacy if the service is insecure, even if it is being self-hosted. This is also especially important considering that it is a "personal" cloud, which should be secure enough to safely store any sensitive personal data that users store on it.

1.2 Quantitative Goals

Depending on use case, some quantitative metrics may be required or at least welcomed by the consumer.

Reliability of the service is crucial, and users' data should remain accessible and loss-proof at all times. The entire point of a personal cloud is storage, and if the service is unreliable and data is inaccessibly or lost, then it is a complete failure of the system.

Performance is use case and demployment-specific, but the software and protocols should introduce minimal performance overhead and be capable of running on low-end consumer machines, e.g. NUCs. Data storage is now a relatively simple task, especially at the basic consumer level, so poor performance should be a cause for concern.

1.3 Use Cases

To what end a user would like to utilise the system may widely vary. However, we will introduce few use cases for the NSS-cloud system.

The personal cloud can be simply used as a storage and sharing platform as any other widely used platform (Google, iCloud etc.). However, due to its locality and low amount of users the consumer benefits by relatively high privacy guarantees and potentially much higher perfomance (HW dependant). Of course for the cost of greater availability outside of the local network. The tradeoff can be somewhat balanced by self hosting the system, which on the other hand increases the attack surface.

Another use could be seen by combination of the personal cloud with a home assistant (e.g. open source Home Assistant), which would have similar benefits as above mentioned use case. Additionally this would lower chance of external misuse and improved responsiveness and debugging of the home assistent settings.

One rather complex, DIY-style, application could be seen with development of custom security cameras. That could potentially be even connected to the previous use case. One could recycle old smartphone to serve as a security camera feed (e.g. over TLS) to the home server, where for example with framework like

OpenCV the images are processed for object detection or individuals identification. The personal cloud could play various roles in this model, from data processing, storing to interfacing such model using its API, and mitigating the thread of possibly sensitive data leakage.

2 System architecture

2.1 Nextcloud server architecture

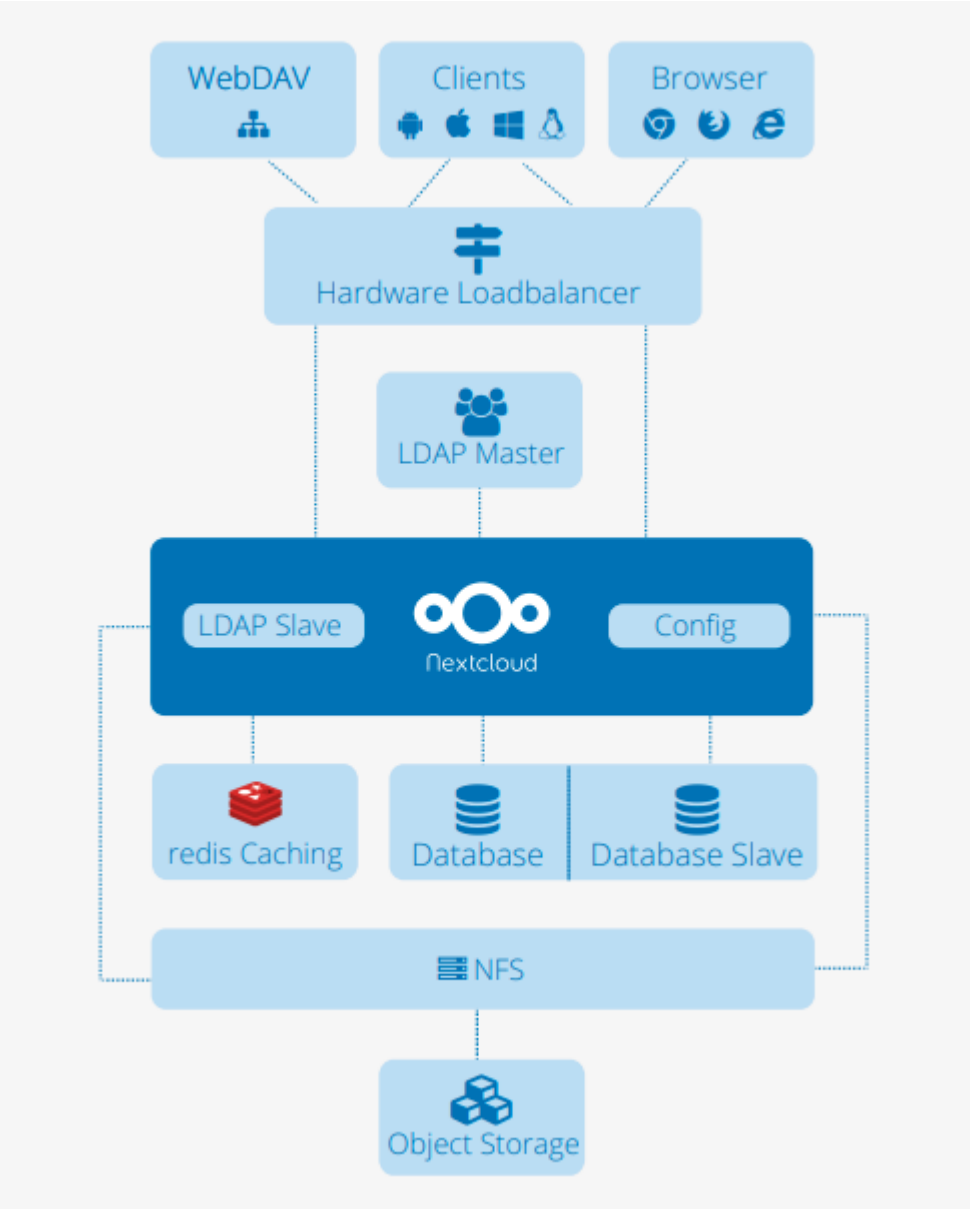

Figure 1 - Scalable Nexcloud setup example. NFS (Network file system) as storage layer, an LDAP (Lightweight Directory Access Protocol) user directory, caching, databases and load balancer. [NC-WP]

In principle Nextcloud server is a web application based on PHP and can be run with any webserver, such as Apache or NGINX. The core of Nextcloud provides storage of file sharing information, user details, application data, configuration and file information in database (MySQL, MariaDB and PostegreSGL are supported). Additional features are available with added services and setup. Plenty of NFS soulutions work natively with Nextcloud. [NC-WP]

To access data, but also to configure, monitor and manage, Nextcloud provides intuitive interface through web browser, Android, iOS or desktop applications. Alternatively resources can handled via WebDAV

standard API.

2.2 NSS-cloud adoption

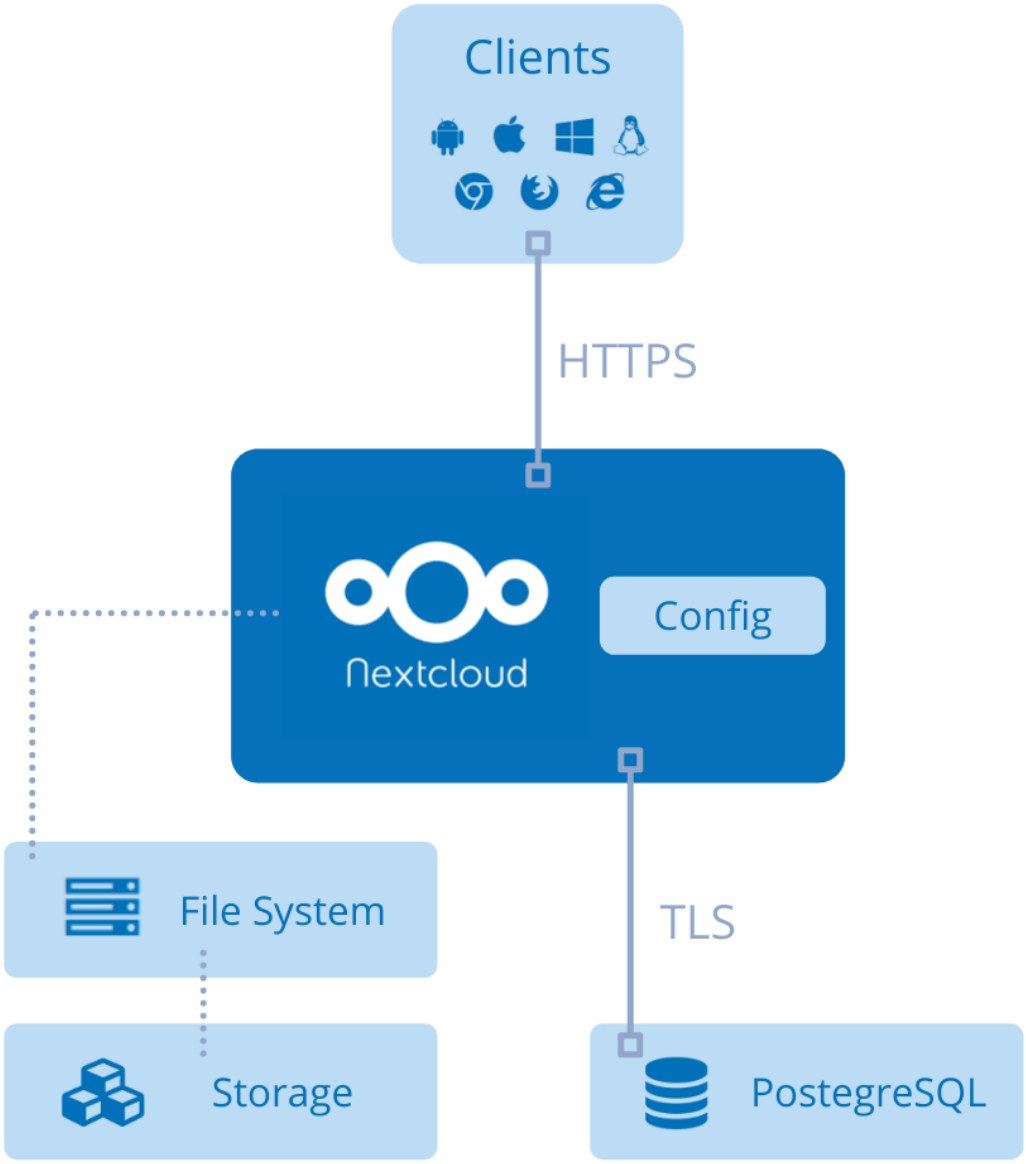

Figure 2 - NSS cloud architecture. Clients connect to Nextcloud over HTTPS. Default OS filesystem is used. PostegreSQL running in a container utilised over TLS.

Since our primary goal is a personal use of the cloud infrastructure (see 1.3 Use Cases) we do not expect high traffic environment and the whole system should comfortably fit on low spec consumer hardware (e.g. NUC). Therefore our practical application has been concluded as depicted in Figure 2.

The Nextcloud server system runs within a single virtual machine (VM). The VM is CSC cPouta standard.small [Pouta-Flavors] flavor. It has 2 vCPUs and 2GB RAM allocated to it and it contains a minimal default installation of Ubuntu Server 20.04 operating system. It is natively installed and configured (not in a container). The system utilises PostgreSQL database which is running inside a docker container hosted by the VM. Communication from and to the PostgreSQL server is done over TLS. The database has restricted

resources access with 384MB for storage (additional 384MB swap space) and limited to upmost 0.5 vCPU usage. Linux native file system interface is used for object storage.

Clients can connect using either the Nextcloud app available on major OS or via web browser interface. The connection is secured over HTTPS and utilises trustworthy automated Let's Encrypt certificate.

For administration purposes the VM exposes a port for secure SSH connection.

```
- The server system runs within a single VM CSC cPouta [aa](www) (Ubuntu
20.04, 2 vCPU, 2GB RAM)
- The Nextcloud has been natively installed and configured
- It utilises PostegreSQL which is running in constrained environment of a
container
    - 384MB storage with 384MB additional swap space from vHDD and allowed
to use upmost 0.5 vCPU
    - the communication protocol between Nextcloud and database is TLS 1.3
(TLS_AES_256_GCM_SHA384)
- Clients connect through web browser using secure HTTPS connection with
Let's Encrypt certificate
```

3 Components

```
Components / Module description including the interfaces exposed between
the modules
```

All in VM, and with Apache. Exposed ports for HTTPS and SSH connections.

3.1 Nextcloud

Direct installation. (nextcloud docs)

TLS 1.3 to database

HTTPS to clients

3.2 PostegreSQL

Database in a docker container connected via TLS 1.3. The ip that Nextcloud is using is force in the configuration to use only TLS. So there is no possiblilty that the connection would not be encrypted. The certificate is also from Lets Encrypt and are copied during Docker init from diskdrive to PostgreSQL configuration directory and also TLS is put on with commandline option.

SSL must be used if the database is not on the same server as the Nextcloud instance, which is not currently the case with our project.

3.3 LDAP

TBC

It will be running inside Docker-container and shall use only encrypted communications.

3.4 Certification BOT (Certbot by EFF)

Let's Encrypt certificate maintainer.

The Certbot updates the cetrificate when necessary. It doesn't directly communicate with the rest of the system and only contacts Let's Encrypt for the cetrificate and then saves the certificate to Apache for the nextcloud to use and for PostgreSQL.

3.5 Clients

Connection over HTTPS, web browser and android app tested.

4 Communication

```
Communication channel between the modules. For instance, do the modules use
secure communication to communicate with each other, if yes, how?
```

Already hinted in previous chapter

- server and client over HTTPS
    - how
- nextcloud and database over TLS (TLS_AES_256_GCM_SHA384)

5 Open source modules evaluation

```
Pros and cons of the open-source components/modules used for developing the
system, and the modules/components you have built (3 points)
```

Open source is the way forward and any proprietary solution should be heavily build on open source. No need to keep reinventing wheel. (TBC reference based open source rant?)

5.1 Nextcloud

pros:

- free and open source
- beginner-friendly basic setup
- can run very lightweight (e.g. on a RasPi) but also scale very large
- modular, can be extended with own or third-party extensions
- decent documentation
- auditability - serves the privacy and security qualitative goal, as an end user can audit the system themself without requiring to put trust in third parties

cons:

- As everyone can see the code and put up their own systems, attacking it is easier. Attacks can be planned and tested without ever touching the target. For example some vulnerabilities might be found easier through reading the code and you don't have to worry about being found out when testing attacks, if you attack a system you yourself set up.

5.2 Let's Encrypt

pros:

- free
- very simple and quick
- only requires a short script to use

5.3 PostgreSQL

pros:

- free and open source

cons:

- not explicitly recommended for use with Nextcloud

5.4 Optional nss-ca

certificate chain generation and signing script. Based on widely used OpenSSL.

- good enough for personal usage
- openssl and crypto in general is not easy and user friendly
- cannot be trustworthy otherwise

Bla bla bla OpenSSL crap and ambiguous documentation etc.

6 Fallacies

```
Which of the fallacies of the distributed system does your system violate,
and how?
```

6.1 Network is reliable

- check if nextcoud mitigades e.g. upload interrupts / session resumption etc (I'd assume so since basic HTTPS config generally supports this on lower levels)

6.2 Latency is zero

- personal network (local), we do not care so much about the latency
- may be considered for self hosting somewhere and implementing e.g. WebRTC based audio/video chat

6.3 Infinite bandwidth

- This should be less of a problem for personal use
- TBC

6.4 Network is secure

Our system secures client communication with HTTPS and internal with TLS. Further consideration could be made to run the system in HSM based secure VM (e.g AMD-SEV/SNP, Intel TDX or Arm CCA in the future) or shifting the system to a container rather than VM and bootstrapping it with HSM (Intel SGX, RISC-V Keystone, TPM 2.0 etc.). This would secure internal communication and compute to the extent that even a physical access to the system would not reveal any sensitive information*.

*We're aware of side channel attacks vulnerabilities of mentioned systems and their are out of the scope of this project.

6.5 Topology doesn’t change

- TBC

6.6 There is one administrator

Nextcloud support multiple administrators, and further specified roles can be delegated to support customized administration topology. [NC-WP]

The underlying OS (Ubuntu 20.04) can be as well configured to support multiple users with various administrative privileges. **(Some Reference here)**

6.7 Transport cost is zero

- TBC (I think a. for our use case of local network we can more or less assume this, b. in Finland the internet access is almost free so zero *additional/solution specific* costs for our consumers?)

6.8 Network is homogeneous

The Nextcloud and our solution supports various communication protocols and APIs.

7 Further development

```
What needs to be added to your system be used to be integrated/extended by
another system.
```

7.1 Deployment

Refactor needed

Despite personal use as main target environment, it would be nice to have some kind of automated deployment script (Especially when things go sideways). From our experience the setup can be done quite quick, but it still is tedious and requires time and certain level of expertise. Definitely not for non-IT user / consumer.

For improved deployability all system components used could be containerized. This would enable automated and centralized deployment which in turns offers better easy of deployment and migration.

Nextcloud maintains stable docker container configuration, which is a good place to start.

7.2 Federation

People on different servers can share files together. Though people still need to log into their own server, this can help extending the system and make communication between users of different servers possible.

7.3 Additional Services and Features

Nextcloud provides user friendly *app store* where once can pick from many available services (e.g. something something, voice channel)

Further WebDAV standard complient API gives an opportunity to independently create custom services if one is of coding nature.

Data storage connection to NAS-like system or at least raid-0 configuration of the system.

7.X Others

- cetrificate: Cetrificate is needed for using https. For example Let's Encrypt is a service that can provide a cetrificate for an ip address. We used Certbot from EFF to get the cetrificate.
- Restricted resources: The `standard.small` VM flavor (type) our project uses in the CSC Pouta has somewhat limited computing resources. To ensure that all the system components this project consists of were allocated adequate amount of memory and CPU cycles we ended up utilizing docker limitations.
    - Docker accepts command line arguments to impose soft- and hard- memory limits and restrictions on CPU resources. The PostgreSQL container was graced with 384 MB with additional 384 MB available for swapping. CPU cycles utilization was constrained to atmost 50% of one of the cores on the VM.
- We ran into problems with MariaDB and Nextcloud co-operation. The problem that rose when trying to connect to database was "Error while trying to initialise the database: An exception occurred while executing a query: SQLSTATE[HY000]: General error: 4047 InnoDB refuses to write tables with ROW_FORMAT=COMPRESSED or KEY_BLOCK_SIZE." We tried to debug it, but after some time with google and Nextcloud errordatabase we decided it would be easier to replace MariaDB with PostgreSQL. And thus we ended up using POstgreSQL instead of MariaDB.

8 Evaluation

```
Methodology used for evaluating the system performance, and the key results
```

- hardware resource usage (idle and under load)
- latency, throughput / bandwidth?
- user experience testing

9 Conclusion / Learning

Nextcloud feels like a great DIY cloud playground.

Resources

[NC-WP] - Nextcloud Solution Architecture whitepaper. Link. Accessed 09.10.2021. [Pouta-flavors] - Virtual machine flavors and billing unit rates. cPouta documentation. - Link. Accessed 12.10.2021