Sync-Polen
A sync&share service for the Portuguese scientific community

...

Mário David (david@lip.pt)
Jorge Gomes, João Pina

Introduction

POLEN: Set of services by the Digital Services Unit (FCCN), for Open Research Data provided for researchers by the national funding agency (FCT).

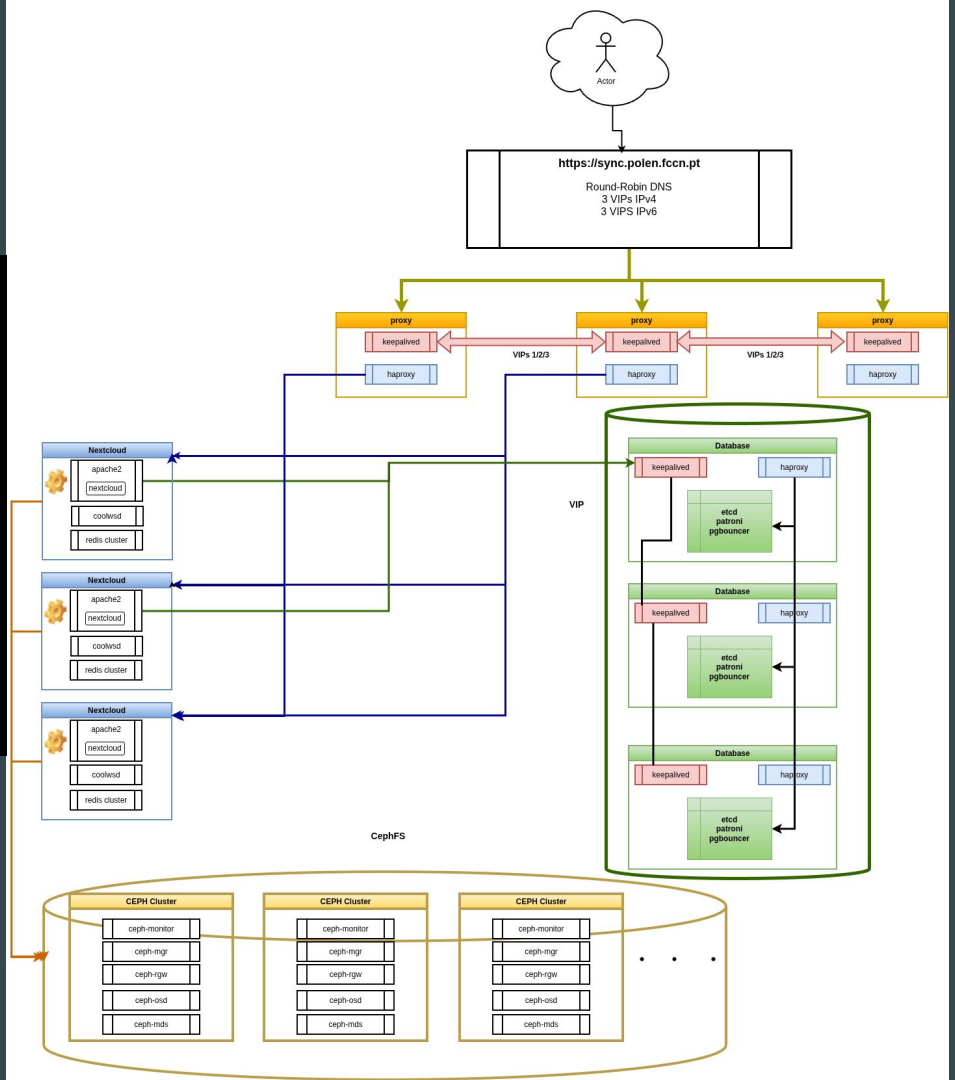
- POLEN Blueprint: Planning of Research Data Management activities throughout the data lifecycle - based on OpenCDMP.
- **POLEN Sync** : Secure and reliable storage space for sharing and synchronizing research data - based on Nextcloud. In pilot phase, but in production infrastructure.
- POLEN DataHub: Sharing and publishing data resulting from research projects - based on Dataverse.

POLEN sync: (Some) Requirements

- The service is sync&share.
- Allow collaborative edition of files (xxOffice).
- Allow authentication with federated identity (CIÊNCIA ID the authentication and identification system for the national Science ecosystem).
- Users should be automatically enabled if they are members of an approved (FCT) project.
- Users are included in a given group that maps one to one with the project.
- Allow the production of metrics such as number of users, number of files, etc.
- The service should be secure implementing best practices.
- The service should have horizontal scalability, load balancing and high availability.
- Implement regular backups of data/files and databases.

Architecture

```
host sync.polen.fccn.pt
sync.polen.fccn.pt is an alias for
sync.a.acnca.pt.
sync.a.acnca.pt has address 193.136.75.226
sync.a.acnca.pt has address 193.136.75.227
sync.a.acnca.pt has address 193.136.75.225
sync.a.acnca.pt has IPv6 address
2001:690:2150:aa:93::6
sync.a.acnca.pt has IPv6 address
2001:690:2150:aa:93::5
sync.a.acnca.pt has IPv6 address
2001:690:2150:aa:93::7
```

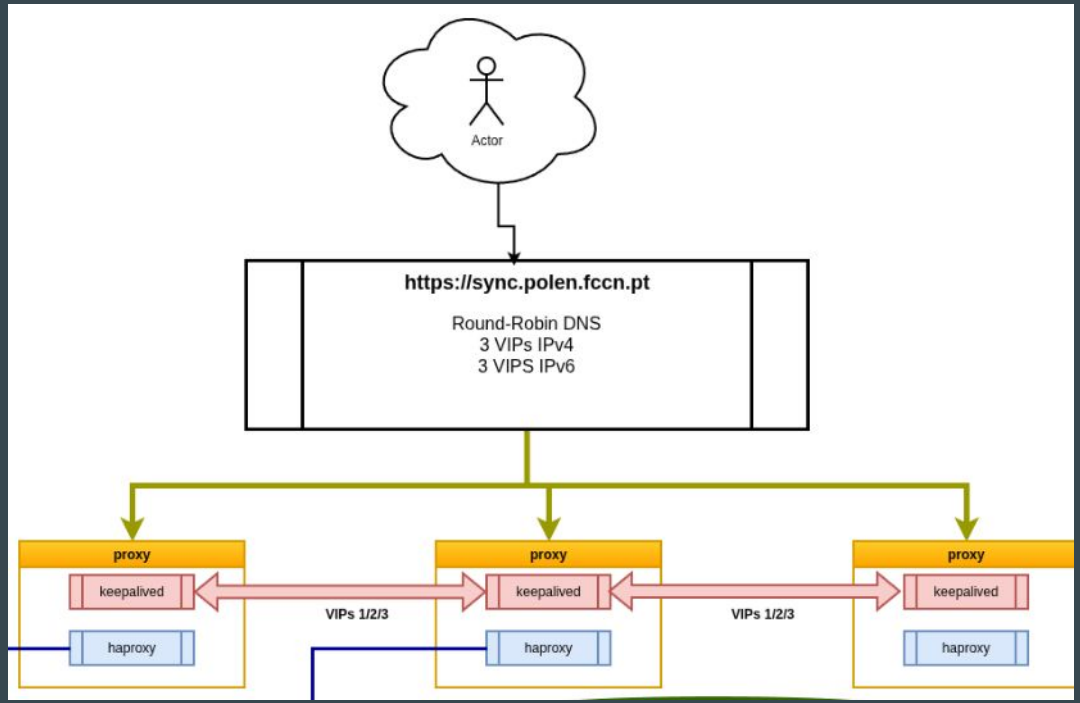


POLEN sync: Platforms

- **Pilot platform** deployed in Openstack cloud infrastructure (with VMs).
  - Simplified architecture:
    - Only 2 HAproxy/keepalived hosts
    - 2 Mariadb hosts in master/slave setup
  - Nextcloud application hosts and use of CephFS the same as in production.
- **Production platform** :
  - Using 3 baremetal nodes with NVMEs.
  - Deployed with LXC/LXD containers (unprivileged mode)
  - CephFS is mounted on the baremetal nodes and bind mounted in LXC/LXD containers.

Architecture I

- 3 proxy nodes:
  - Keepalived - VIP (**sync.polen.fccn.pt**)
  - Haproxy - to the backend application, terminates SSL
  - Round-robin DNS:
    - 3 VIPs IPv4
    - 3 VIPs IPv6




```
unicast_src_ip 10.93.3.221
unicast_peer {
  10.93.3.222
  10.93.3.223
}

virtual_ipaddress {
  193.136.75.225
}
```

Keepalived.conf - use of unicast avoids the need to use a Public IP for the host (similar for IPv6 section)

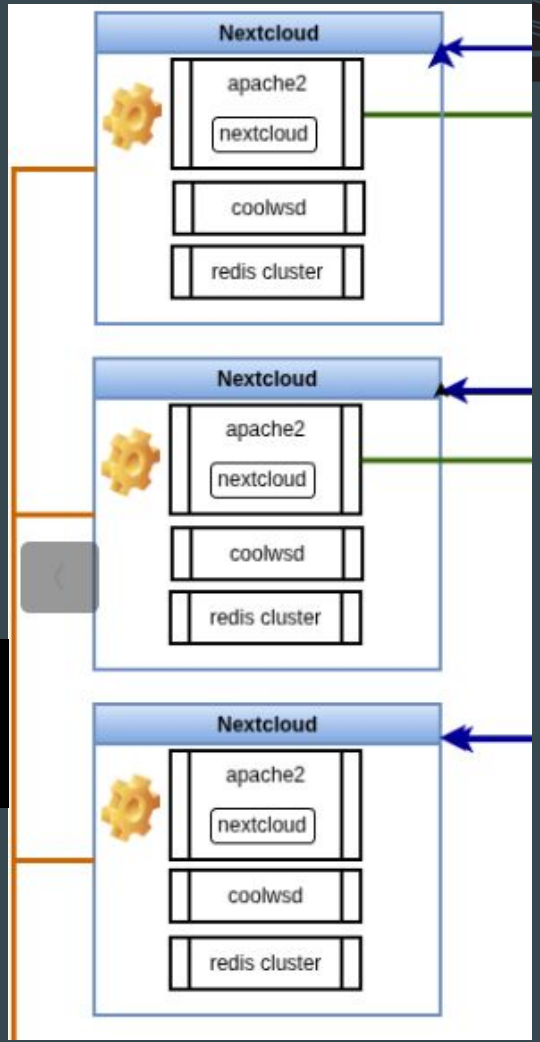
Architecture II

- 3 application nodes:
  - Are the backend of HAproxy
  - Run:
    - Apache2
      - Nextcloud application
    - Coolwsd (Collabora)
    - Redis cluster

```
polen_sync@<CephFS_ID>.polen_sync=/  500T  413G  500T   1% /var/www

# Nextcloud application  -> /var/www/nextcloud
# Data/Users files       -> /var/www/data
```

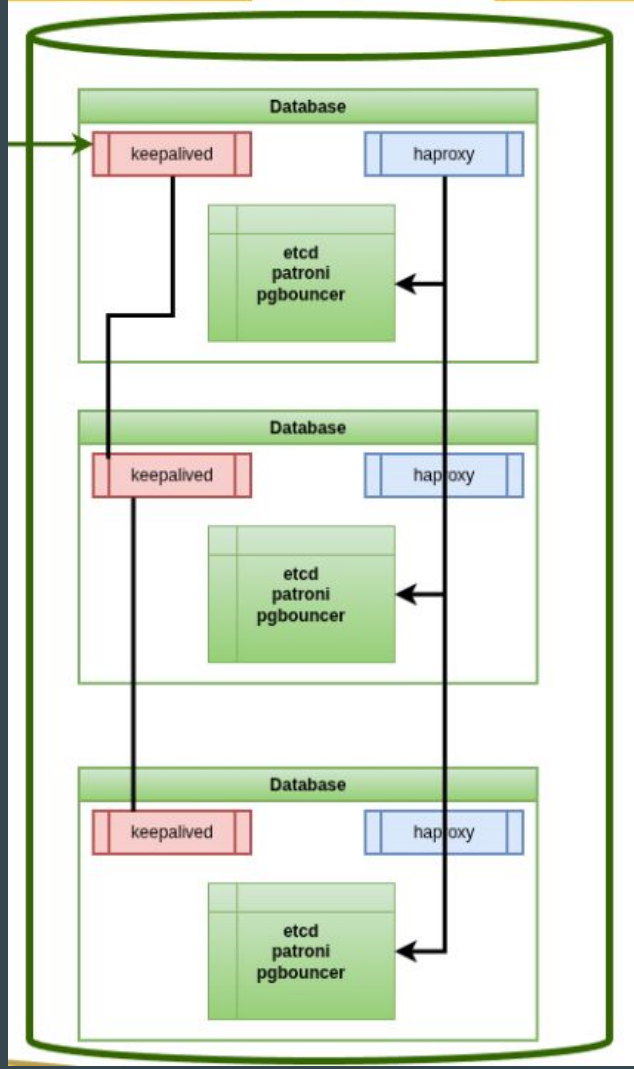
- Nextcloud application directory in the shared CephFS between the three nodes.
- Installation and upgrades done on only one of the nodes.
- Coherency regardless of node you land.



Architecture III

3 nodes for the database → Postgresql cluster.

- **keepalived** : 1 VIP, 1 node is MASTER 2 are BACKUP
- **haproxy** : balances between the three **pgbouncer** on each node.
- **pgbouncer** : lightweight, high-performance **connection pooler** for PostgreSQL.
- **etcd** : A distributed, consistent key-value store:
  - Acts as the Distributed Configuration Store (DCS) for **Patroni** .
  - Only stores cluster metadata, dynamic configurations, and the **primary leader key/lease** .
- **patroni** : manages the Postgres service:
  - Provides a command interface for node administration and monitoring.

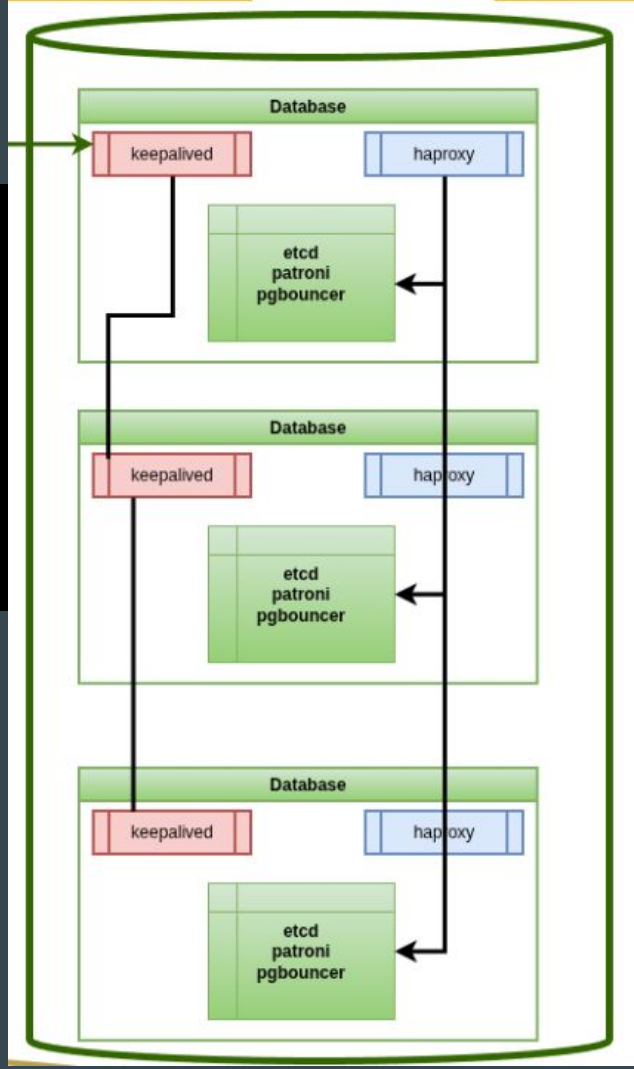


Architecture IV

```
Client/Nextcloud DB connection: [VIP]:5000

keepalived [VIP]: 10.0.0.1 ->
    -> haproxy frontend [VIP]: 10.0.0.1:5000 ->
        -> haproxy backend [pgbouncer]: 10.0.1.1:6432 |
                                        10.0.1.2:6432 |->
                                        10.0.1.3:6432 |

            -> pgbouncer.ini -> patroni(postgre): 10.0.1.1:5432
```



Ceph cluster

There are 2 types of hosts:

- 64 hosts with HDDs (49 deployed)
- 28 hosts with SSDs, from which:
- 3 hosts with SSDs for the services

The Operating System is Ubuntu 24.04LTS

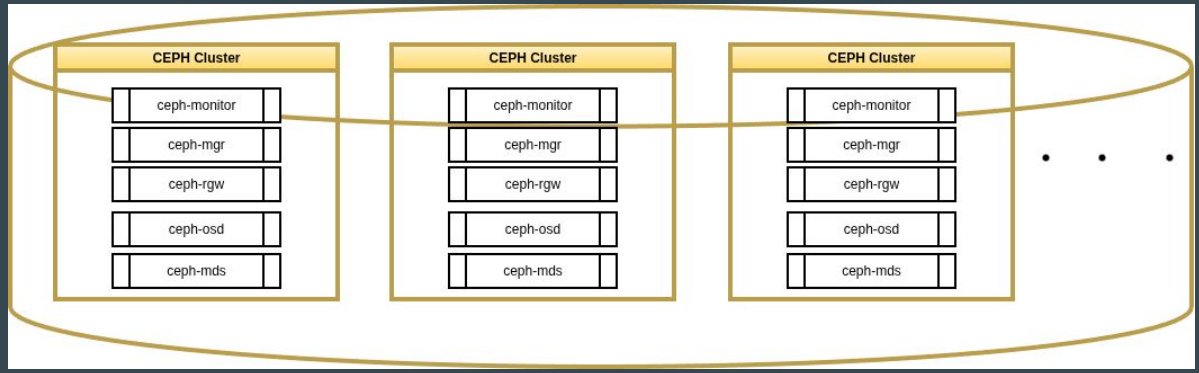
Ceph version: Tentacle 20.2.2

Deployed with podman containers

Ansible playbooks

Manual cluster initialization

CephFS for underlying storage; mounted on the 3 application nodes.



HDD hosts, the devices are as follows:

- 10 HDD devices of 18.2 TBytes each.
- 2 SSD devices of 5.8 TBytes each.
- SSD devices -> 5 partitions each that will serve as `Write Ahead Log - WAL` and Bluestore RocksDB for each of the 10 HDD devices that will hold the data (OSDs).

```
--- RAW STORAGE ---
CLASS     SIZE
hdd     9.2 PiB
ssd     1.3 PiB
TOTAL    10 PiB
```

Nextcloud installation and configuration

Done with ansible playbooks:

1. 00-nc-install.yml - installs necessary packages depending on the type of node.
2. 01-nc-redis-cluster.yml - deploys and start the redis cluster.
3. 02-nc-db.yml - 02_2-nc-postgres.yml - Deployed and configures the full Postgresql cluster and creates the database.
4. 03-nc-conf.yml - copies all configuration files.
5. 04-nc-haproxy.yml - configures keepalived and haproxy.
6. 05-nc-nextcloud.yml - deploys and initialize the Nextcloud service
    a. Uses ansible-galaxy collection nextcloud.admin.
    b. Allows the use of occ nextcloud CLI for initial installation and further configuration of the config.php, or installation of more apps.

The Nextcloud directory and data directory are in the CephFS mounted on all three application nodes. In this way the initialization is done in one of the nodes, and the same config and deployment on all 3 nodes.

Nextcloud initialization

Ansible playbook snippets from 05-nc-nextcloud.yml

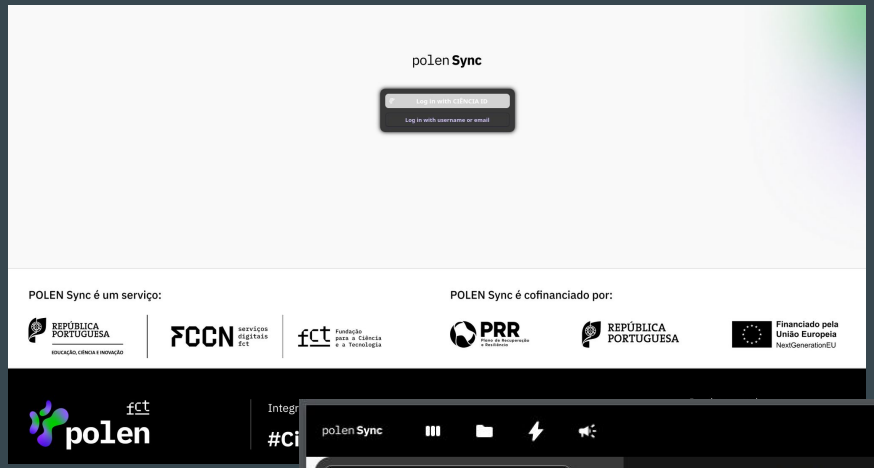
```yaml
- name: Check Nextcloud CAN_INSTALL exists.
  ansible.builtin.stat:
    path: "{{ nextcloud_webroot }}/config/CAN_INSTALL"
  register: nc_nextcloud_can_install
```

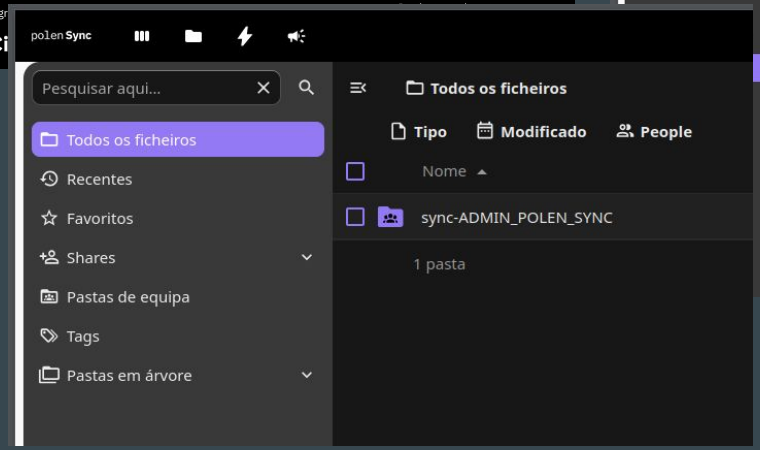
```yaml
- name: Run occ installation command      You, 8 months
  become: true
  nextcloud.admin.run_occ:
    nextcloud_path: "{{ nextcloud_webroot }}"
    command: >-
      maintenance:install
      --database "mysql"
      --database-host "{{ mysql_master_ip }}"
      --database-name "nextcloud"
      --database-user "nextcloud"
      --database-pass "{{ vault_db_nextcloud_pass }}"
      --admin-user "admin"
      --admin-pass "{{ vault_nc_admin_pass }}"
  when: nc_nextcloud_can_install.stat.exists
```

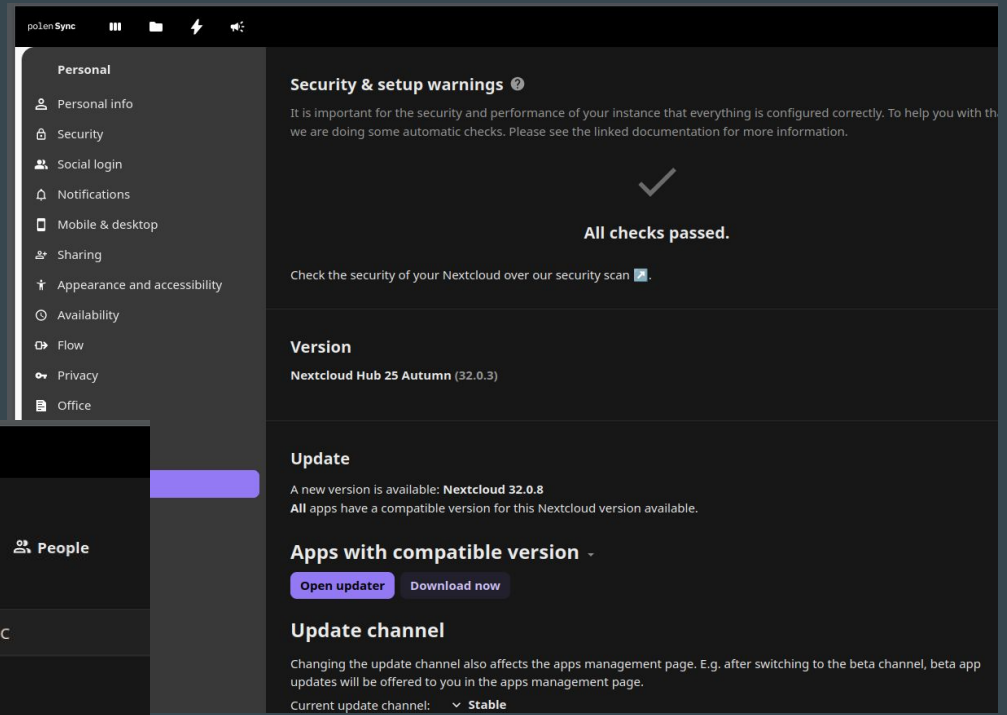
```yaml
nc_config_settings:
  - { name: 'auto_logout', value: true }
  - { name: 'datadirectory', value: '{{ nc_datadir }}' }
  - { name: 'default_phone_region', value: 'PT' }
  - { name: 'default_timezone', value: 'Europe/Lisbon' }
  - { name: 'enable_previews', value: false }
  - { name: 'htaccess.RewriteBase', value: '/' }
  - { name: 'log_type', value: 'file' }
  - { name: 'log_type_audit', value: 'file' }
  - { name: 'logfile', value: '/var/log/nextcloud/nextcloud.log' }
  - { name: 'logfile_audit', value: '/var/log/nextcloud/nextcloud.log' }
  - { name: 'loglevel', value: '1' }
```

```yaml
- name: Set Nextcloud settings in config.php
  become: true
  nextcloud.admin.run_occ:
    nextcloud_path: "{{ nextcloud_webroot }}"
    command: config:system:set {{ item.name }} --value {{ item.value }}{% if item.value | type_debug == 'bool' %} --type=boolean{% endif %}
  with_items:
    - "{{ nc_config_settings }}"
```

https://sync.polen.fccn.pt







Availability and destructive tests I

All tests performed in the pilot infrastructure.

Failover haproxy

The two haproxy+keepalived hosts should cover each other. The premises are:

1. If one host goes down: the VIP should migrate to the other host through the keepalived service.
2. If the haproxy service stops or has some problem and stops working: the VIP should migrate to the other host through the keepalived service.

Use rclone (WebDAV protocol) to download and upload a significant amount of files and directories to the nextcloud platform:

1. When the download and the upload was undergoing, we have shutdown the haproxy host that had the VIP.
2. When the download and the upload was undergoing, we have stopped the haproxy service:
    a. `systemctl stop haproxy` on the host that had the VIP.

In both situations, we observed **NO** noticeable impact on the transfers, neither the download nor the upload.

We observed that the VIP had migrated to the other host.

Availability and destructive tests II

Failover nextcloud host

There are three nextcloud application hosts, they are under the backend of the haproxy hosts. To test the failover and resilience to the failure of 1 or 2 nodes, we have made a similar test as the previous one. (Used rclone to download and upload a significant amount of files and directories to the nextcloud platform).

When the download and the upload where ongoing :

- We have stopped the apache2 service in one node, and then also in another node leaving just one node with the apache running, we have started again apache2 in the stopped node, and several combinations always leaving one running.

During this process we were checking the transfers either upload or download. In all situations, we observed **NO** noticeable impact on the transfers.

Availability and destructive tests III

Total loss of platform

There is a dump of the mariadb nextcloud database , also the ***data and the nextcloud application itself are located in the CephFS*** .

For this test we have made a mysqldump of the nextcloud database and copied to another host.

We have done a terraform destroy - removes all resources, and then we have done a terraform apply to instantiate again all resources.

Execute the disaster recovery procedure...

Disaster recovery procedure

The procedure is almost the same as when installing and configuring for the first time: Execute all playbooks except **05-nc-nextcloud.yml**

At this point we restore the previously dumped database in the database master node.

Execute the last playbook to update any needed remaining nextcloud configuration.

The last playbook can be executed anytime we need to update the nextcloud configuration, it does not destroy or initializes the application anew.

Operational matters

- Scripts and cron jobs that were developed to help operations and satisfy several requirements.
- In all cases, we use directories located in the CephFS, thus preserving all files and directories in case something happens to the platform.
- The rate and retention policies in each case where agreed upon with FCCN.

Operational matters: Scripts and Cron jobs I

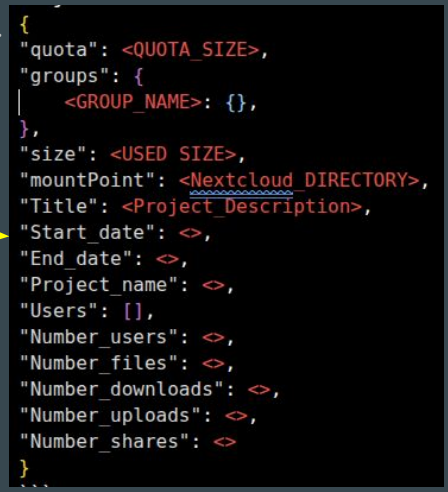
- *Database backups* :
  - A script performs a dump of the nextcloud database every **3 hours** , through a cron job.
  - Backups are kept for **3 months** .
  - This cron job is only executed in one of the Nextcloud nodes.
- *Consolidation of logs* :
  - The nextcloud logs of all three nodes are located in /var/log/nextcloud, a script executes a rsync of this directory to one directory in CephFS through a **daily** cron job.
  - These logs are kept for **1 year** .

Operational matters: Scripts and Cron jobs II

- ***Automatic creation of nextcloud group folders:***
  - A systemd service executing a script, was implemented with the objective of creating a new nextcloud folder whenever a new group is created.
  - When the user logs in, he sees this group folder and is only allowed to create and access files and folders in this top folder.
- ***Get approved research projects from FCCN service API (called Sciproj)*** :
  - A script gets a dump of all projects in Sciproj in json format.
  - Executed every **week** through a cron job on only one of the nextcloud nodes.
  - It stores this json in a directory located in the CephFS.
- ***Backups*** :
  - A script performs a **daily** backup of the CephFS directory to a S3 object store, located in the Lisbon site, through a cron job on only one of the nextcloud nodes.

Operational matters: Scripts and Cron jobs III

- ***Management of projects and email notifications*** :
  - A shell script produces json files of Nextcloud groupfolders, groups, users and a csv of usage report (from the app usage-report. It's executed daily by the cron job.
  - These json files serve as input to a python script (also execute by the above mentioned shell script.
  - The python script has 3 objectives:
    i. Construct a **json** with all information of all projects.
    ii. e-mail notifications:
      - notify users users that the project is going to end the next day.
      - notify users they are going to lose access to the project, 4 months after the end of the project.
      - notify users the project and all its files are going to be removed, 6 months after the end of the project.
    iii. Monthly accounting usage: a csv with accounting usage is sent every month by e-mail to the service administrators.



Security

Most of it from Nextcloud security hardening:

- Service exposed as “https/ssl” - the haproxy terminates it.
- fail2ban on haproxy and nextcloud nodes.
- Nextcloud web admin interface is restricted to local network.
- Enabled HTTP Strict Transport Security.

Security auditing

- Performed by FCCN experts
- Black-box testing, using tools such as; Wappalizer, Penetration testing kit, nmap, Nessus, OWASP, XSStrike, sqlmap, etc.
- Found:
  - 2 medium impact issues
  - 2 low impact issues

Security auditing: medium impact issues

1. Public exposed ports:
    a. ssh port 22: we restricted to the local network
    b. Collabora service on port 9980: we put ACLs in haproxy to expose only port 443:
        i. acl coolwsd path_beg -i /browser /hosting/discovery /hosting/capabilities /cool
2. The following endpoints are publicly exposed:
    a. /remote.php
    b. /status.php
    c. On this matter we have done nothing since these endpoints are an essential part of Nextcloud functionality.

Security auditing: low impact issues

1. Vulnerable JS dependency: We have not acted on this since it is part of the application dependencies and eventually will be tackled by the developers.
2. Apache2 Server Leaks Version Information: We have set the following options in the server:

```
cat conf-enabled/security.conf

ServerTokens Prod

ServerSignature Off

TraceEnable Off

Header set X-Content-Type-Options: "nosniff"

Header set Content-Security-Policy "frame-ancestors 'self';"
```

Summary, future

Security audit was very useful, permitted hardening the service.

We opened the production infrastructure to users in the beginning of last September.

We will soon pass from pilot to production phase

We have currently: 17 groups/projects and 37 users.

We plan to execute several performance and benchmark tests on the platform

Thanks