Author	Ilia Kebets
Reviewer	Thiébaud Modoux
Date	04.06.2019
Version	1

Pryv.io core migration

Procedure

Summary

We copy the data from the old core to the new one then set the old core to proxy to the new one so we can use it during the DNS propagation phase.

(optional) create user(s) with specific data on source for post-migration verification

Generate a few events and streams by hand for a naked eye comparison for data transferred after the migration. The easiest way would be to use the web app https://sw.${DOMAIN}/access/signinhub.html to create a few events, streams and images (attachment files).

Deploy and launch services on the destination machine

We assume that core is already deployed (config present, docker images downloaded) on the *dest* machine.

Launch services by running `${PRYV_CONF_ROOT}/run-core` and verify that all containers are started using `docker ps` and check logs on `core`, `preview`, `hfs` containers.

Transfer user data

User data migration has a down time which we'll call *cold* migration. To limit its duration, we transfer the bulk of the data from *source* to *dest* prior to the *cold* migration using `rsync`. The *cold* migration consists of syncing the most recent data changes. After this, services will be started on *dest* and the `nginx` process on *source* will proxy calls while DNS entries are updating.

1. Create an SSH key pair using `ssh-keygen -t rsa -b 4096 -C "migration@remote"` and copy the private one to `${PATH_TO_PRIVATE_KEY}` in *dest* and add the public one in `authorized_keys` on *source*.
2. Transfer Mongo data: on *dest*, run `time rsync --verbose --copy-links --archive --compress --delete -e "ssh -i ${PATH_TO_PRIVATE_KEY}" ${USERNAME}@${SOURCE_MACHINE}:${PRYV_CONF_ROOT}/core/mongodb/data/ ${PRYV_CONF_ROOT}/core/mongodb/data/`
3. Transfer attachments data: on *dest*, run `time rsync --verbose --copy-links --archive --compress --delete -e "ssh -i ${PATH_TO_PRIVATE_KEY}" ${USERNAME}@${SOURCE_MACHINE}:${PRYV_CONF_ROOT}/core/core/data/ ${PRYV_CONF_ROOT}/core/core/data`
4. If needed, Repeat steps 2-3 to sync the biggest bulk of the data prior to the *cold* migration
5. Shutdown services on *source*: `${PRYV_CONF_ROOT}/stop-containers`
6. Make last sync by executing steps 2-3

If you wish to reactivate service on the *source* machine, simply reboot the stopped services: `${PRYV_CONF_ROOT}/run-core`

Set NGINX proxying

Since the DNS changes will take some time to come into effect, the NGINX process on *source* will be set to proxy to the *dest* machine. The following steps describe the configuration changes to make the *source* NGINX proxy calls to the *dest* core. It is advised to comment out the old setting inline using # in order to rollback easily in case of need.

- In `${PRYV_CONF_ROOT}/core/nginx/conf/site-443.conf`, Replace the following:

```
    upstream core_server {
      server core:3000 max_fails=3 fail_timeout=30s;
      server core:3001 max_fails=3 fail_timeout=30s;
      server core:3002 max_fails=3 fail_timeout=30s;
      server core:3003 max_fails=3 fail_timeout=30s;
      server core:3004 max_fails=3 fail_timeout=30s;
      server core:3005 max_fails=3 fail_timeout=30s;
    }

    upstream websocket_server {
      ip_hash;
      server core:3000 max_fails=3 fail_timeout=30s;
      server core:3001 max_fails=3 fail_timeout=30s;
      server core:3002 max_fails=3 fail_timeout=30s;
      server core:3003 max_fails=3 fail_timeout=30s;
      server core:3004 max_fails=3 fail_timeout=30s;
      server core:3005 max_fails=3 fail_timeout=30s;
    }

    upstream hfs_server {
```

```
    server hfs:3000 max_fails=3 fail_timeout=30s;
    server hfs:3001 max_fails=3 fail_timeout=30s;
    server hfs:3002 max_fails=3 fail_timeout=30s;
    server hfs:3003 max_fails=3 fail_timeout=30s;
    server hfs:3004 max_fails=3 fail_timeout=30s;
    server hfs:3005 max_fails=3 fail_timeout=30s;
  }

  upstream preview_server {
    server preview:9000 max_fails=3 fail_timeout=30s;
  }
```

with

```
  upstream core_server {
    server ${DEST_CORE_IP_ADDRESS}:443;
  }

  upstream websocket_server {
    server ${DEST_CORE_IP_ADDRESS}:443;
  }

  upstream hfs_server {
    server ${DEST_CORE_IP_ADDRESS}:443;
  }

  upstream preview_server {
    server ${DEST_CORE_IP_ADDRESS}:443;
  }
```

In the same file, change proxy protocol from `http` to `https`

- Change: `http://core_server` to `https://core_server`
- Change: `http://websocket_server` to `http://websocket_server`
- Change: `http://hfs_server` to `https://hfs_server`
- Change: `http://preview_server` to `https://preview_server`

Run `${PRYV_CONF_ROOT}/run-core`

As we are currently using docker-compose to specify the mounted volumes (containing the NGINX config), we just boot all services, even if they will be ignored as NGINX is proxying to the *dest* machine.

Verify

Log onto an account and verify that the data has been moved. You can monitor the services logs to ensure that data is accessed on the new machine.

Update core server on Register

Update the core IP address in the register machines. In ${PRYV_CONF_ROOT}/reg-{master/slave}/dns/dns.json, update this value: `dns.staticDataInDomain.${CORE_SUBDOMAIN} = { "ip": ${DEST_CORE_IP_ADDRESS}}` and reboot the services.