Author	Thiébaud Modoux
Reviewer	Ilia Kebets (v3)
Date	08.05.2019
Version	4

Pryv.io emails

Install and configure the sending of emails

Summary

Pryv.io allows to send emails in two situations:

- Account creation
- Password reset requests

This document will guide you through the configuration of the sending of these emails.

This requires a basic understanding of SMTP servers, SPF records and DNS zone settings.

This document covers:

1. How to configure Pryv.io core(s) to send welcome or password reset emails
    1. Using Mandrill
    2. Using the Pryv.io mail microservice
2. How to install and configure the Pryv.io mail microservice
    1. Installation and deployment variants
    2. Configuration
3. How to set the right SPF records for the Pryv.io associated domain

This document does not cover:

- How to obtain or build the SPF record for your domain
- How to set the SPF record on a domain other than the one used by your Pryv.io platform

1. Configure Pryv.io core(s) to send welcome or password reset emails

1.1. Using Mandrill

As a first solution, you can use the Mandrill email API.

The cores' configuration file should contain the following:

```
# core/core/conf/core.json

  "services": {
    ...
    "email": {
      "enabled": {
        "welcome": true,
        "resetPassword": true
      },
      "method": "mandrill",
      "url": "https://mandrillapp.com/api/1.0/messages/send-template.json",
      "key": ${apiKey},
      "welcomeTemplate": "welcome-email",
      "resetPasswordTemplate": "reset-password"
    }
  }
```

Notes:

- Replace `${apiKey}` with the API key of your own Mandrill account.
- The templates names (`welcome-email`, `reset-password`) have to correspond to templates you have created within your Mandrill account.
- The `enabled` flags allow to enable/disable granularly the sending of emails according to their type.
- For now, the Pryv.io integration with Mailchimp does not support languages.
- **WARNING**: this solution may not be compliant with privacy regulations.

The Mandrill configuration will require you to set a SPF record in the DNS zone of the domain you wish to use to send those emails. See the SPF section for details about these records.

If you choose to use Mandrill, no further configuration is needed, so you can skip the rest of this document.

1.2. Using the Pryv.io mail microservice

A second solution is to use the Pryv.io generic and containerized mail microservice.

This microservice will either send emails using a SMTP server under your control or using its internal sendmail binary, as explained in the Configuration section.

It uses customizable email templates, which can be written in different languages, such as HTML or pug.

The cores' configuration file should contain the following:

```
# core/core/conf/core.json

  "services": {
    ...
    "email": {
      "enabled": {
        "welcome": true,
        "resetPassword": true
      },
      "method": "microservice",
      "url": ${microserviceUrl},
      "key": ${sharedKey},
      "welcomeTemplate": "welcome-email",
      "resetPasswordTemplate": "reset-password"
    }
  }
```

Notes:

- Replace `${sharedKey}` with a symmetric key that will also be set within the mail microservice (as shown in the Configuration section).
- The templates names (`welcome-email`, `reset-password`) have to correspond to template folders that exist in the mail microservice.
- The `${microserviceUrl}` is the URL on which the mail microservice is receiving mailing requests, so it depends on your deployment choice. See the section below to complete the configuration.

2. Install and configure the Pryv.io mail microservice

2.1. Installation and deployment variants

The Pryv.io dockerized mail microservice can be deployed according to various infrastructure setups.

In this section, we present three suggested deployment variants. You can adapt and combine these setups, see the Deployment variants chapter below.

We recommend to deploy the microservice on a different machine than the core to avoid giving an access to the personal data to the people working on the templates.

Microservice URL

We suggest to define a TYPE A entry in the Pryv.io DNS pointing to the machine where the microservice is deployed, thus using `https://mail.${DOMAIN}` as `microserviceUrl`.

As explained in the DNS config documentation, this is done by adding the following entry:

```
# reg-master/dns/conf/dns.json

  "dns": {
```

```
    ...
    "staticDataInDomain": {
      ...
      "mail": {
        "ip": "${MAIL_SERVICE_MACHINE_IP_ADDRESS}"
      }
    }
  }
```

Variant 1: Host on the same machine as register

The first suggestion is to deploy the mail microservice on the master register machine. This is the variant we recommend, so you can find a complete example in our template cluster configuration.

First, add the mail Docker image to register's Dockerfile:

```
# reg-master.yml

  mail:
    image: "pryvsa-docker-release.bintray.io/pryv/mail:1.1.3"
    networks:
      - frontend
    volumes:
      - ./reg-master/mail/conf/:/app/conf/:ro
      - ./reg-master/mail/templates/:/app/bin/templates/:ro
    restart: always
```

Notes:

- The mail configuration file is located in `/reg-master/mail/conf/mail.json`.
- The template folders and files are located in `/reg-master/mail/templates/`.
- See the Configuration section below for more details about configuring the service.

Then, in each core's configuration file, replace the `${microserviceUrl}` as follows:

```
  # core/core/conf/core.json

  "services": {
    ...
    "email": {
      ...
      "url": "https://mail.${DOMAIN}/sendmail/",
    }
  }
```

You will need to add the following in the register's NGINX config file:

```
# reg-master/nginx/conf/site.conf

upstream mail_server {
  server mail:9000;
}

# Mail server
server {
  listen               443 ssl;
  server_name          mail.${DOMAIN};
  access_log           /app/log/mail.log;

  client_max_body_size  5M;

  location /sendmail/ {
```

```
        proxy_pass http://mail_server;
      }
    }
```

Pros:

- No additional machine is required
- SSL termination is available
- Template updates do not require access to sensitive user data

Cons: Template updates require access to the register machine

Variant 2: Host on an external machine

The mail microservice can also be deployed on a machine external to Pryv.io. This setup is similar to the register one and requires a NGINX container for SSL termination.

Pro: Template updates can occur without giving access to any user data

Cons: Requires an additional machine

Variant 3: Single-node deployment

This is not recommended outside of development environments. Thus said, you can find a complete example in our single node template configuration.

You need to add the mail service in the docker-compose `pryv.yml` file as done in the other variants.

The difference here is that core does not require to send the requests passing through the public internet and can do it locally.

Then, in core's configuration file, replace the `${microserviceUrl}` as follows:

```
    # pryv/core/conf/core.json

    "services": {
      ...
      "email": {
        ...
        "url": "http://mail:9000/sendmail/",
      }
    }
```

Pro: No need of additional machine

Cons: Template updates require access to the single Pryv.io machine, which contains sensitive user data

Deployment variants in details

Here are some criteria for choosing the right setup for your deployment of the Pryv.io mail microservice:

- Mail templates are located on the storage of servers operating the microservice. To modify the templates, a procedure of publication should be put in place. In some situations, access to servers such as service-core or service-register might be too sensitive and requires the mail microservice to be deployed on a separate server.

- Codes enabling reset of passwords will transit by the mail microservice. This is considered as very sensitive information. You might want to lower the risk of exposure by having the mail microservice installed on a closed loop or directly on service-core(s) (this is also a possible reason for prefering the mail microservice over Mandrill).
- Limiting the number of machines deployed is also a factor of choice. Thus using the same machine as the one hosting service-register could be a good option for some installations.

This can lead to the following setup options (changing from one option to another is possible):

1. Single separate mail server
    - **Pro**: 1 single place with all the templates and confined access
    - **Cons**: 1 more machine, 1 single point with all the reset codes transit, requires SSL termination set-up
2. Deploy within register (simplest)
    - **Pro**: 1 single place with all the templates, SSL termination is ready
    - **Cons**: 1 single point with all the reset codes transit, a procedure to update templates on the external machine (register) should be established
3. Using separated servers on the same network
    - **Pro**: confined access to templates
    - **Cons**: N more machines, requires SSL termination set-up
4. Deploy within cores
    - **Pro**: SSL termination ready
    - **Cons**: a procedure to update templates on sensitive machines (cores) should be established

2.2. Configuration

The Pryv.io mail microservice can be configured by providing a configuration file (.json, .hjson or .yaml) containing settings that we explain below.

Templates

Templates consist of pug files, arranged into folders according to email types and langage codes, such as:

```
/mail
└── /templates
      |
      └── /welcome-email
      |     └── /en
      |     |     html.pug
      |     |     subject.pug
      |     └── /fr
      |           html.pug
      |           subject.pug
      |
      └── /reset-password
            └── /en
            |     html.pug
            |     subject.pug
            └── /fr
                  html.pug
                  subject.pug
```

The default root folder for templates can be changed by providing **templates.root**.

If the template for the requested language does not exist, the service will try to find another template for the same email type but with a default language (e.g. english instead of french). The default language can be defined in configuration by providing **templates.defaultLang**.

Transport

The Pryv.io mail microservice allows to define two types of transport: SMTP or sendmail command.

SMTP

SMTP transport is used by default, it allows to define an external mail delivery service through configuration:

- **smtp.host**: SMTP host (e.g. smtp.ethereal.email)
- **smtp.port**: SMTP port (e.g. 587)
- **smtp.auth.user**, **smtp.auth.pass**: credentials to authenticate against an external mail service (e.g. sendgrid)

Sendmail

An alternative is to use the sendmail command, which is installed within the mail microservice container.

It has to be explicitly activated through configuration:

- **sendmail.active**: true

HTTP

Within the HTTP parameters, we define the IP address and port on which the mailing server will be listening.

Also, we set here the `${sharedKey}` we previously defined within the core configuration.

- **http.ip**: 0.0.0.0 (docker container localhost)
- **http.port**: 9000
- **http.auth**: `${sharedKey}`

Full settings

Here is a sample configuration that shows all available settings alongside with some explanation:

```
# mail/conf/mail.json

{
  // Logging settings
  logs: {
    prefix: '',
    console: { active: true, level: 'info', colorize: true },
    file: { active: false },
  },
  email: {
    message: {
      // Sender name and email address
      from: {
        name: "Ethereal Email",
```

```
            address: "btvryvs5al5mjpa3@ethereal.email"
          }
        }
      },
      // By default, the the mail microservice will use SMTP as transport
      smtp: {
        // SMTP host of the external email delivery service
        host: "smtp.ethereal.email",
        // SMTP port
        port: 587,
        // Credentials to authenticate against SMTP server
        auth: {
          user: "btvryvs5al5mjpa3@ethereal.email",
          pass: "VfNxJctkjrURkyThZr"
        }
      },
      // Alternative transport, using the sendmail command of the machine
      sendmail: {
        // Sendmail transport takes precedence over SMTP transport if both are set to true
        active: false,
        // Default location where the sendmail command is installed, within the mail microservice container
        path: '/usr/sbin/sendmail'
      },
      http: {
        // IP address on which the mailing server is listening
        ip: "0.0.0.0",
        // Port on which the mailing server is listening
        port: 9000,
        // Each sendmail request should contain authorization header that matches this key (used to prevent abu
        auth: "CHANGEME",
      },
      templates: {
        // Root folder where the templates are stored, mounted as volume within the container
        root: '/templates/',
        // Default language for templates
        defaultLang: 'en'
      }
    }
```

3. SPF records for the Pryv.io domain

SMTP servers use SPF records to help prevent email spoofing. In order to send an email on behalf of a certain domain, you will need to add the SPF record associated with your SMTP server to your domain's DNS zone.

If you choose to use domain associated with your Pryv.io platform, you should add a SPF record similar to this one:

> @ 10800 IN TXT "v=spf1 include:spf.mandrillapp.com ~all"

In the SPF record above, we declared that Mandrill can be used to send emails on behalf of the domain of our Pryv.io platform. You can of course replace Mandrill by the SPF address of the SMTP host(s) of your choice.

Please refer to the DNS config documentation on how to set such SPF record in the Pryv.io DNS.