ProtonMail Bridge

Use ProtonMail Bridge to interact with your ProtonMail account through the nice Sailfish E-Mail app. If you have any questions, post them here: https://forum.sailfishos.org/t/protonmail-bridge-on-sailfish/6017

Using ProtonMail Bridge in an LXC container

Step 1: Create a LXC container

To create and manage an LXC container, you can use the app "Containers" which is available from Chum. After launching the app, you can create a new container by tapping on the "+" icon. This brings up a page where you give your container a name and decide on one of the available Linux-distros and the corresponding version. This will be setup for you automatically. The "Architecture"-field can (normally) left at the default architecture. Do not check "Setup desktop" as this is not required. You can choose any distro you like as we will use the Nix package manager to install ProtonMail Bridge. Why? Because Proton doesn't provide an *aarch64*-build of Bridge and compiling it from source (in my experience) often doesn't work correctly.

> **ATTENTION:** If you want to use NixOS itself in an LXC container then there are special steps required to run ProtonMail Bridge. Instructions for these steps can be found here

Step 2: Install the Nix package manager

After the container has been created by the app, you need to attach a terminal session to it. This can easily be done by clicking on the container you created and clicking on the button "Attach". This will open a new terminal window where you can execute commands *inside* your container. If you have a terminal session attached, the Nix package manager can be installed by executing (in the attached session):

```
sh <(curl -L https://nixos.org/nix/install) --daemon
```

(more info here: https://nixos.org/download)

> **Note:** Make sure *curl*, *tar* and *xz* are installed for this script to work. If you get an error that it can't find some package, just install the package it says is missing

Step 3: Installing the necessary programs

Next, we will need to install a few programs (with Nix). Obviously, we will need ProtonMail Bridge itself which can be installed with

```
nix-env -iA nixpkgs.protonmail-bridge
```

Then, we will also need pass and screen. These can be either installed via your distro's package manager or via Nix:

```
#Install screen:
nix-env -iA nixpkgs.screen
#Install pass
nix-env -iA nixpkgs.pass
```

Step 4: Setup Bridge

> **Disclaimer**: For this step, I'm following this tutorial, where you can find more information if you want

Step 4a: Setting up pass

Generate a PGP key pair for the new user with an empty passphrase.

The empty passphrase is required to run ProtonMail Bridge on the background on system startup without being prompted for the password and hence causing the process to fail.

```
gpg --full-generate-key
```

Choose 1 for RSA, type in 2048 for the length, choose 0 to make the key not expire, type in your name and your e-mail address, leave comment and passphrase empty (you will be asked for it two times).

Next, you can initialize pass. This can be done by executing

```
pass init <E-Mail address>
```

You need to use the same E-Mail address you typed in previously when generating the key.

Step 4b: Setting up Bridge

For Bridge to work, you will need to execute

```
export HOME=/home/root
```

(the directory must exist, of course).

After that, you can launch Bridge with:

```
protonmail-bridge --cli
```

You are now in the ProtonMail Bridge command line interface. Here, you can add a new account by typing in *add*. This will prompt you for your username and password (and Two-Factor-Token if enabled). After that, it will begin synchronising which can take a while. You than can list the credentials for IMAP and SMTP by typing in *info*. These details you can now use to setup the Jolla E-Mail app. The server URL is *127.0.0.1*. You can type in exit to quit Bridge.

> **Note**: closing the terminal window and/or Container-app does NOT stop the container, so it can conveniently run in the background and receive e-mails without you even noticing.

Step 5: Make Bridge automatically launch (optional)

If you want to start Bridge automatically, you can follow a few simple steps (some things again taken from here)

First, create a basic script which will launch ProtonMail Bridge in the background:

```
mkdir /var/lib/protonmail
nano /var/lib/protonmail/protonmail.sh
```

In this file, you can copy the content from here. At the top of the script (after the shebang *#!/bin/bash*), you need to add the *export*-statement we used before.

After that, you need to create a systemd-service that starts and stops ProtonMail Bridge with the container. This can be done with:

```
nano /etc/systemd/system/protonmail.service
```

In there, copy the content from here

All you have to do now is to enable the service with

```
#Enable autostart
systemctl enable protonmail
#Start the service manually
systemctl start protonmail
```

Now ProtonMail Bridge will start every time you start your container. If you want, you can also configure the app "Containers" to automatically launch after every boot with the app "Takeoff" available from OpenRepos)

LXC Container configurations for ProtonMail Bridge

This chapter collects pages on which alternative configurations, tips and tricks are provided how to get ProtonMail Bridge running on SFOS - while there is no native daemon available. Please read first the introduction page of this book to get started.

NixOS as LXC Container for ProtonMail Bridge

Tested on/with:

- SFOS 4.5.0.24 on Xperia 10 III
- Harbour Containers 0.8.1-1.3.1.jolla
- lxc-templates-desktop 1.4.-1.1.1.jolla

Pre-requisites:

- Remote login on the SFOS with *devel-su* rights (I would not try the below with *fingerterm* ...)
- Fresh SFOS instance using the "*Distribution*" (tempate) "*nixos*" from Harbour Containers
  - Give it a name "Bridge" (of course, you can give whatever name you want, but below we use "Bridge")
  - Start "Bridge" using the Harbour Containers to see if there is no show stopper for the below for some reason.
  - Now you can close the Harbour Containers, is is not needed below, but is a useful tool when on the road you want see if the instance is running or not.

Connection to the NixOS instance

```
[root@Xperia10III defaultuser]# lxc-attach -n Bridge
[root@nixos:/]#
```

Configuration files you need to create or edit

The editor is `nano` - you do not need that for long, in the configuration files you can get more editors, here `vim` will be installed.

```
[root@nixos:/]# ls -al -R /etc/nixos
/etc/nixos:
total 24
```

```
drwxr-xr-x  3 root root 4096 Jan  1 22:34 .
drwxr-xr-x 17 root root 4096 Jan  5 06:34 ..
-rw-r--r--  1 root root 1104 Jan  1 22:34 configuration.nix
-rw-r--r--  1 root root  126 Dec 29 22:07 lxd.nix
-rw-r--r--  1 root root  693 Dec 31 22:42 protonmail-bridge_service.nix
drwxr-xr-x  2 root root 4096 Jan  1 22:46 scripts

/etc/nixos/scripts:
total 12
drwxr-xr-x 2 root root 4096 Jan  1 22:46 .
drwxr-xr-x 3 root root 4096 Jan  1 22:34 ..
-rwxr--r-- 1 root root  813 Jan  1 22:46 protonmail.sh
```

Note that execution bit for the `protonmail.sh` `(chmod 744 protonmail.sh` when you are at that point) - otherwise the automatic start as service would not work. You need to create the `scripts` directory and the script in it.

Other files you need to create are `lxd.nix` and `protonmail-bridge_service.nix`.

configuration.nix

```
[root@nixos:/etc/nixos]# cat configuration.nix
# Edit this configuration file to define what should be installed on
# your system.  Help is available in the configuration.nix(5) man page
# and in the NixOS manual (accessible by running â€˜nixos-helpâ€™).

{ config, pkgs, lib, modulesPath, ... }:

{

  imports =
    [
      # Include the default lxd configuration.
      "${modulesPath}/virtualisation/lxc-container.nix"
      # Include the container-specific autogenerated configuration.
      ./lxd.nix
      ./protonmail-bridge_service.nix
    ];

  environment.systemPackages = with pkgs; [
    vim
    binutils
    protonmail-bridge
    gnupg
    pinentry-curses
    pass
    screen
    # Define a package for the Proton Bridge service script
    (pkgs.writeShellScriptBin "protonmailBridgeScript" (builtins.readFile /etc/nixos/scripts/protonmail.sh))
  ];

  networking.useDHCP = false;
  networking.interfaces.eth0.useDHCP = true;
  networking.firewall.enable = false;
  services.openssh.enable = false;
  services.dbus.enable = true;

  services.protonmailBridge.enable = true;

  system.stateVersion = "23.11"; # Did you read the comment?
}
```

You can see that we want to install `vim`, `binutils`, `protonmail-bridge` (of course), `gnupg`, `pinentry-curses`, `pass` and `screen`. Of these, the introduction article explains the usage of GPG for the user authentication and the usage of `screen`-package to allow to run the CLI-type of ProtonMail Bridge as a service.

The version above is a stable version 23.11 of NixOS - you can pick yours if you do not like this one from https://nixos.org/channels/.

lxd.nix

```
[root@nixos:/etc/nixos]# cat lxd.nix
{ config, pkgs, ... }:

{
  # Configuration related to LXC containers
  nix.settings.sandbox = false; # Disable sandboxing

}
```

protonmail-bridge_service.nix

```
[root@nixos:/etc/nixos]# cat protonmail-bridge_service.nix
{ config, pkgs, lib, ... }:

let
  cfg = config.services.protonmailBridge;
in
{
  options.services.protonmailBridge = {
    enable = lib.mkEnableOption "ProtonMail Bridge";
  };

  config = lib.mkIf cfg.enable {
    systemd.services.protonmailBridge = {
      description = "ProtonMail Bridge";
      after = [ "network.target" ];
      wantedBy = [ "multi-user.target" ];

      serviceConfig = {
        Type = "forking";
        ExecStart = "/run/current-system/sw/bin/protonmailBridgeScript start";
        ExecStop = "/run/current-system/sw/bin/protonmailBridgeScript stop";
        User = "root";  # It's better to use a non-root user
        Restart = "always";
      };
    };
  };
}
```

protonmail.sh

```
[root@nixos:/etc/nixos/scripts]# cat protonmail.sh
#!/bin/bash

export HOME=/root
export PATH=${PATH}:/root/.nix-profile/bin
:
case "$1" in
  start)
    # will create an screen in detached mode (background) with name "protonmail"
    screen -S protonmail -dm protonmail-bridge --cli
    echo "Service started."
    ;;
  status)
    # ignore this block unless you understand how screen works and that only lists the current user's screens
    result=$(screen -list | grep protonmail)
    if [ $? == 0 ]; then
      echo "Protonmail bridge service is ON."
    else
      echo "Protonmail bridge service is OFF."
    fi
    ;;
  stop)
    # Will quit a screen called "protonmail" and therefore terminate the running protonmail-bridge process
    screen -S protonmail -X quit
    echo "Service stopped."
    ;;
  *)
    echo "Unknown command: $1"
    exit 1
  ;;
esac
```

All Set - let's remove NixOS ! (not quite) ...

```
nix-channel --remove nixos
```

... to install the one above

```
nix-channel --add https://nixos.org/channels/nixos-23.11 nixos
nix-channel --update
nixos-rebuild switch
```

How to install or update programs

The below programs should come by the rebuild. If you wonder how they would get otherwise installed or how you can update them at a later date, this is the syntax:

```
nix-env -iA nixos.protonmail-bridge
nix-env -iA nixos.gnupg
nix-env -iA nixos.pinentry-curses
nix-env -iA nixos.pass
nix-env -iA nixos.screen
```

But there is normally no reason for that anymore, but you may like `emacs -nw` more than vim declared above for installation ...

If you arer reading this for updating the `protonmail-bridge` software version at a later date, please stop the `protonmailBridge` service before doing that using the `systemctl` commands explained below.

Test protonmail-bridge CLI manually

The main page of this book will explain you how (see **Step 4**). But before you go, do not forget to add this line in `~/.gnupg/gpg-agent.conf`:

```
[root@nixos:~/.gnupg]# cat gpg-agent.conf
pinentry-program /root/.nix-profile/bin/pinentry-curses
```

If the file does not exist, create it.

Test as system service

The idea is, of course to make the container to work as daemon which you can start and stop to provide ProtonMail Bridge as service for your SFOS native email client or other (see the main page of this book **Step 4b**).

If you have any protonmail-bridge CLI sessions running, quit them now.

```
[root@nixos:/]# systemctl start protonmailBridge
```

```
[root@nixos:/]# systemctl status protonmailBridge
â—� protonmailBridge.service - ProtonMail Bridge
     Loaded: loaded (/etc/systemd/system/protonmailBridge.service; enabled; preset: enabled)
    Drop-In: /nix/store/y5fajbjj3czy2rcqvijkx0s8faf0cgdr-system-units/service.d
             â””â”€zzz-lxc-service.conf
     Active: active (running) since Fri 2024-01-05 23:02:57 UTC; 57s ago
    Process: 270 ExecStart=/run/current-system/sw/bin/protonmailBridgeScript start (code=ex>
   Main PID: 273 (screen)
         IP: 0B in, 0B out
     Memory: 28.4M
        CPU: 750ms
     CGroup: /system.slice/protonmailBridge.service
             â”œâ”€273 SCREEN -S protonmail -dm protonmail-bridge --cli
             â”œâ”€275 protonmail-bridge --cli
             â”œâ”€308 keyboxd --homedir /root/.gnupg --daemon
             â”œâ”€320 gpg-agent --homedir /root/.gnupg --use-standard-socket --daemon
             â””â”€322 scdaemon --multi-server

Jan 05 23:02:57 nixos systemd[1]: Starting ProtonMail Bridge...
Jan 05 23:02:57 nixos protonmailBridgeScript[270]: Service started.
Jan 05 23:02:57 nixos systemd[1]: Started ProtonMail Bridge.
lines 1-20/20 (END)
```

Other ways to check

```
[root@nixos:/]# screen -list
There is a screen on:
        273.protonmail  (Detached)
1 Socket in /tmp/screens/S-root.
```

How to join the protonMailBridge service

This is useful if there is a missing network connection, or if your credentials do not work, you can join the protonmail-bridge CLI to see its messages by writing:

```
[root@nixos:/]# screen -r protonmail
```

Other debugging commands for a failed service

This is quite useful to find the missing executables (or file without execution rights - see above for the script):

```
[root@nixos:/]# journalctl -u protonmailBridge.service --no-pager -n 50
```

If something is missing to run the script, it is perhaps easiest to find the missing command's location (always the linked one in NixOS, not the actual file with cryptic and random directory name) using `which` command and then add that in the `PATH` environment variable of the `protonmail.sh` script.

Scripting in the SFOS side

Forget all the "*user state LXC containers*" and all that - since the prerequisite is that you can be *devel-su*, you can use `sudo`, right?

Example

I have a script which knocks out the SFOS phone for the night and makes it brain dead, or almost. Well, I do not want the newly established ProtonMail Bridge "service" to run without networks or anything, so I stop the LXC container. In the morning I do the inverse (with *Situations*):

```
[defaultuser@Xperia10III situations]$ cat sleep-start.sh
#!/bin/sh
#
sudo /usr/bin/systemctl stop wpa_supplicant >/tmp/stop-wpa_supplicant-root.log 2>/tmp/stop-wpa_supplicant-root.err
#
sudo /usr/bin/systemctl stop bluetooth >/tmp/stop-bt2-root.log 2>/tmp/stop-bt2-root.err
#
sudo /usr/bin/systemctl stop ofono >/tmp/stop-ofono-root.log 2>/tmp/stop-ofono-root.err
#
sudo /usr/bin/systemctl stop connman >/tmp/stop-connman-root.log 2>/tmp/stop-connman-root.err
# No reason to keep e-mail Bridge running over the night, be quiet, too!
sudo /usr/bin/lxc-stop -n Bridge >/tmp/bridgestop.log 2>/tmp/bridgestop.err
#
/usr/bin/systemctl --user stop pulseaudio.service >/tmp/pulseaudiostop.log 2>/tmp/pulseaudiostop.err
#
sleep 5
#
# Make your alarm clock sure to make some noise in the morning
/usr/bin/systemctl --user start pulseaudio.service >/tmp/pulseaudiostart.log 2>/tmp/pulseaudiostart.err
```