NGINX : UTILISER L'AUTH OPENID DE LA PLM

PIERRE GAMBAROTTO

INSTITUT DE MATHÉMATIQUES DE TOULOUSE

Journées Mathrice 20/10/2021

QUI JE SUIS ?

- responsable info IM de Toulouse
- ASR, déploiement reproductible
- Développeur, programmation fonctionnelle
- pas encore devops …

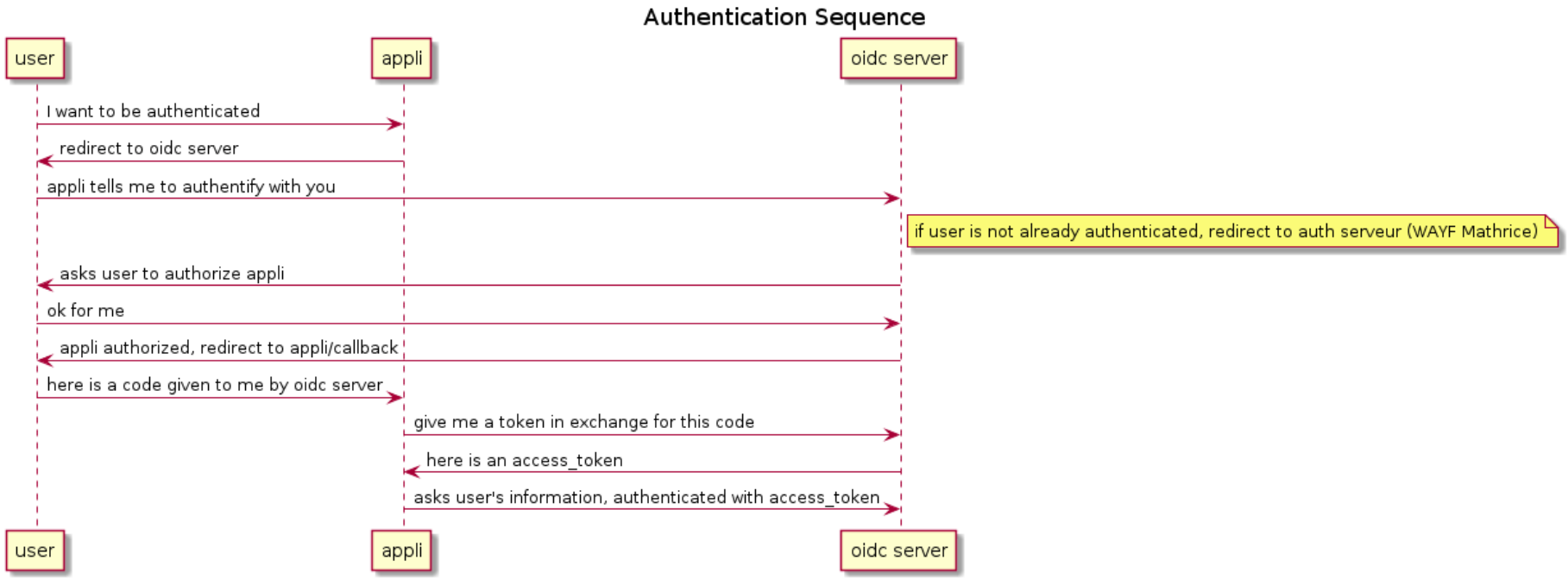
OBJECTIFS

- Authentifier une application web
- Par la couche sso de la PLM
- En en faisant le moins possible …
- Du moins à terme :-)

Retour d’expérience !

OAUTH/OPENID-CONNECT

plmlab:gamba/oidc$_{plmapplication}$



ENREGISTRER L'APPLICATION

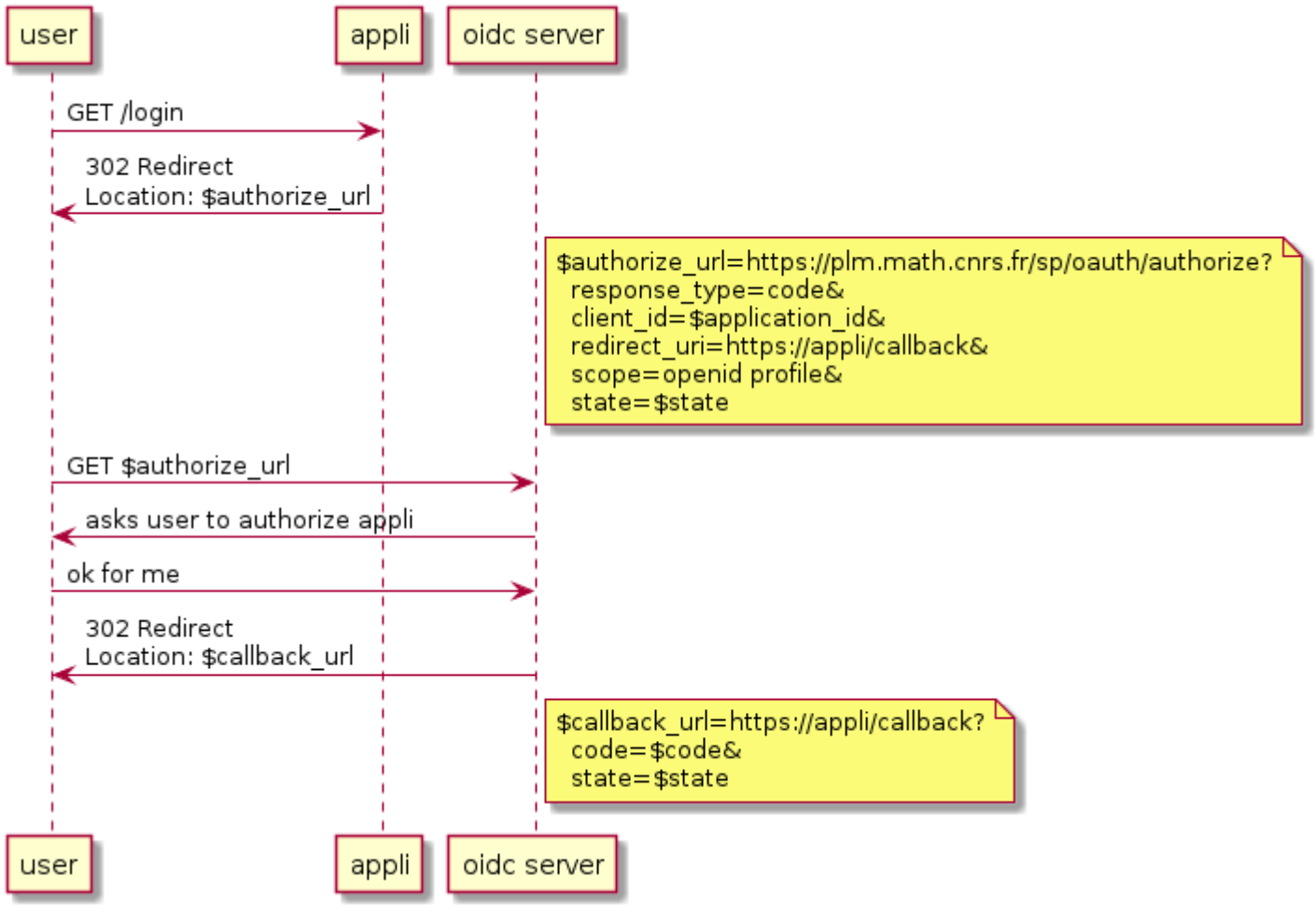
Enregistrement de l'application sur
https://plm.math.cnrs.fr/sp/oauth/applications

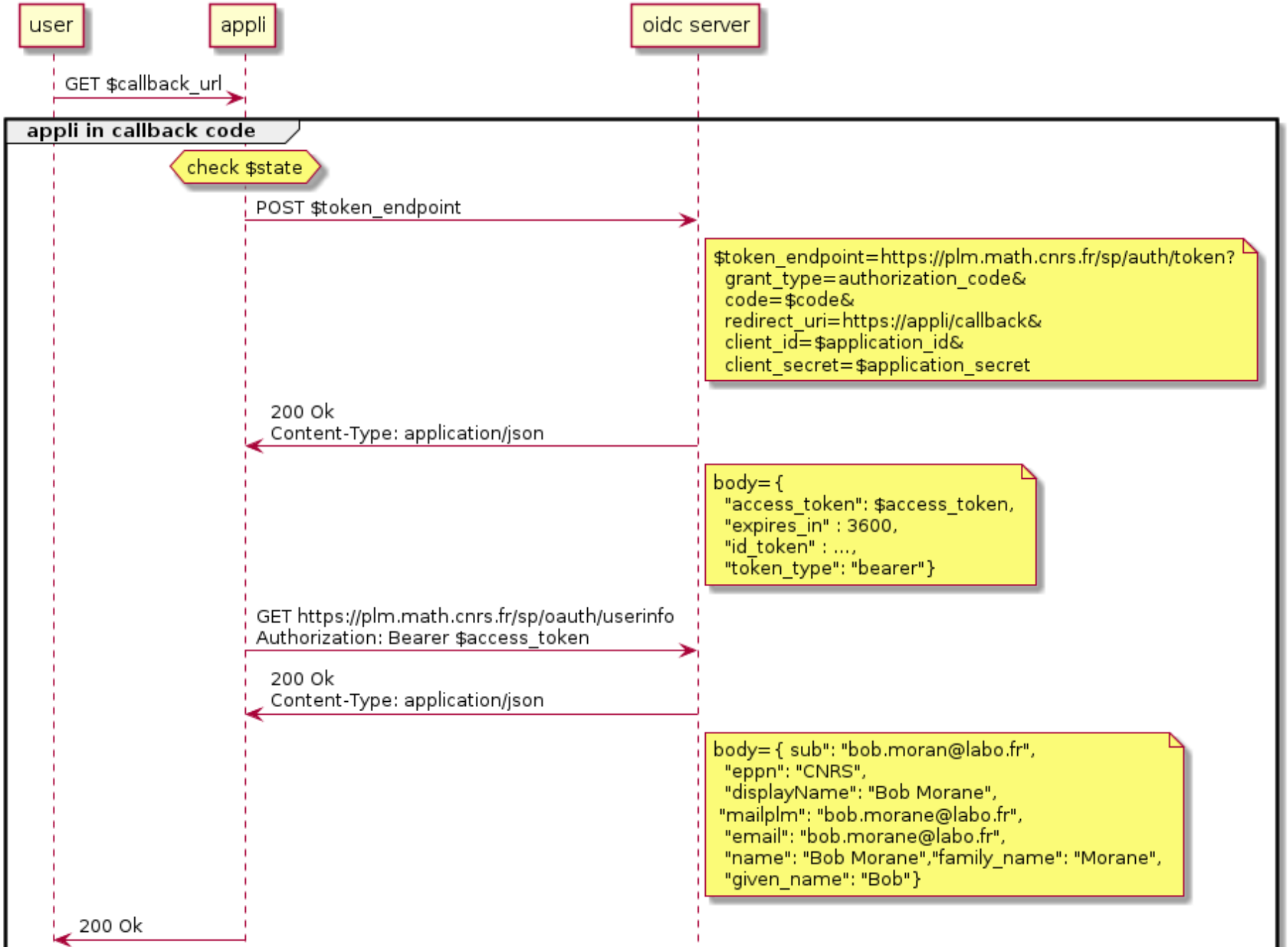
- `application_id`
- `application_secret`
- l'URL de callback, https://appli/callback

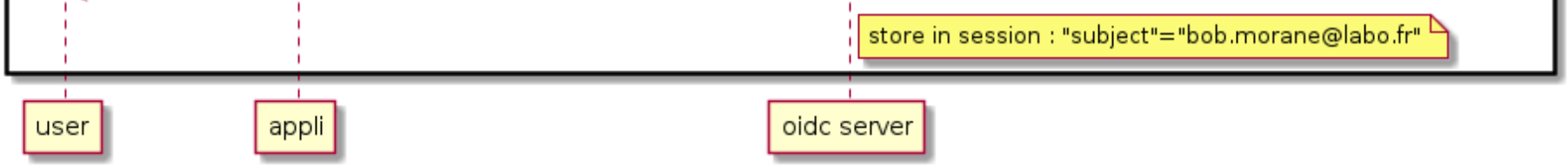
callback : le serveur oidc redirige le navigateur utilisateur APRÈS l'authentification

- peut être `https://localhost:1234/...`
- doit être en `https`

CODER LES FLUX HTTP




user
appli
oidc server
GET $callback_url
appli in callback code
check $state
POST $token_endpoint
$token_endpoint=https://plm.math.cnrs.fr/sp/auth/token?
grant_type=authorization_code&
code=$code&
redirect_uri=https://appli/callback&
client_id=$application_id&
client_secret=$application_secret
200 Ok
Content-Type: application/json
body={
"access_token": $access_token,
"expires_in" : 3600,
"id_token" : ...,
"token_type": "bearer"}
GET https://plm.math.cnrs.fr/sp/oauth/userinfo
Authorization: Bearer $access_token
200 Ok
Content-Type: application/json
body={ sub": "bob.moran@labo.fr",
"eppn": "CNRS",
"displayName": "Bob Morane",
"mailplm": "bob.morane@labo.fr",
"email": "bob.morane@labo.fr",
"name": "Bob Morane","family_name": "Morane",
"given_name": "Bob"}
200 Ok



user

appli

oidc server

À LA MAIN OU AVEC UNE LIBRAIRIE

- gérer la configuration provenant de l'enregistrement
- `GET $SERVER/.well-known/openid-configuration`
  - `authorization_endpoint`
  - `token_endpoint`
  - `userinfo_endpoint`

- générer `authorize_url` : générer un secret, le stocker dans un cookie chiffré
- un flux http à suivre
- gérer tous les cas d’erreurs
- ne pas négliger la sécurité

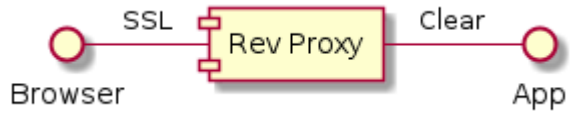
DRY

Don’t Repeat Yourself

1. code à la main
2. librairie
3. infrastructure !

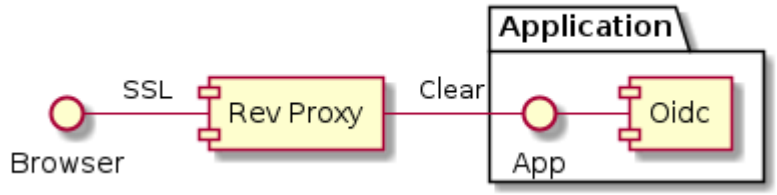
MIGRER DANS L'INFRA

Reverse Proxy : tout le traffic est géré par l'application

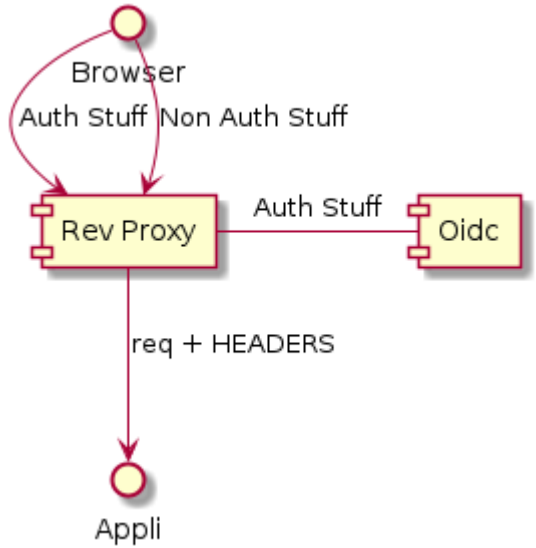


POINT DE DÉPART :

- traffic lié à l’authentification
- traffic normal



OBJECTIF



On déplace dans le reverse proxy la partie de l'application qui assurait le traffic d'authentification.

LES PRODUITS INVESTIGUÉS

Facteurs discriminant : pouvoir exploiter les *claims* imbriqués

claim : tester une sous-partie des infos sur l'utilisateur

Ex :

- `sub == pierre.gambarotto@math.univ-toulouse.fr`
- `sub = /@math\.univ-toulouse\.fr$/`
- `legacyplm.ou contains "1234"`

LES CANDIDATS

- traefik : partie pro seulement
- apache : configuration à base de directive
- oauth2-proxy : en go, à surveiller
- nginx : configuration en lua

NGINX + LUA

raison du choix :

- déjà utilisé pour la terminaison ssl
- expérience passée en lua


WORLD
WARCRAFT
Alliance
Horde
Troll
Shaman
Gender
Skin Color
Face
Hair Style
Hair Color
Tusks
Randomize
Horde
Four races comprise the Horde: the brutal orcs, the shadowy undead, the spiritual tauren, and the quick-witted trolls. Beset by enemies on all sides, these outcasts have forged a union they hope will ensure their mutual survival.
Troll
Once at home in the jungles of Stranglethorn Vale, the fierce trolls of the Darkspear tribe were pushed out by warring factions. Eventually the trolls befriended the orcish Horde, and Thrall, the orcs' young warchief, convinced the trolls to travel with him to Kalimdor. Though they cling to their shadowy heritage, the Darkspear trolls hold a place of honor in the Horde.
Shaman
Shaman commune directly with the elements. Their combination of wisdom and resilience makes them ideal as tribal advisors and leaders. In battle the shaman use totems and spells to manipulate the elements and provoke other fighters to untold heights of rage and might. Shaman exemplify the primal bond between the savage races and their environment.
Name
Randomize
Accept
Back

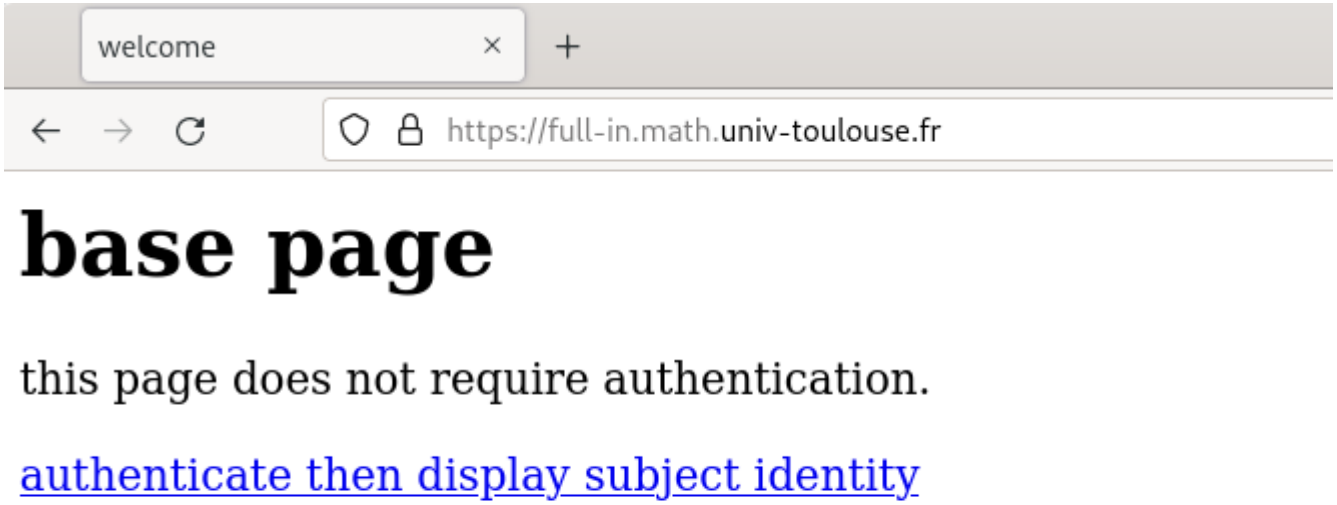
LUA

- langage simple
- embarqué comme langage d'extention dans nginx
- allocation dynamique
- openresty facilite énormément l'environnement avec nginx
- nombreuses librairies : json, oidc, jwt

DÉMO

Pour avoir une idée du flux que l'on veut : https://full-in.math.univ-toulouse.fr/

- page de base, non authentifiée
- lien : https://full-in.math.univ-toulouse.fr/header/X-Sub => partie «interne» du site



accès à une page interne : auth obligatoire

En vous authentifiant, vous acceptez l'utilisation de votre adresse électronique à des fins d'information sur l'état des services de la PLM [en savoir plus].

Sélectionnez ou cherchez l'établissement auquel vous appartenez.

Veuillez sélectionner l'établissement auquel vous appartenez. ... Connexion

☐ Se souvenir de mon choix définitivement et contourner cette étape à partir de maintenant.

Si vous possédez un compte PLM et préférez l'utiliser

Connexion avec un compte PLM

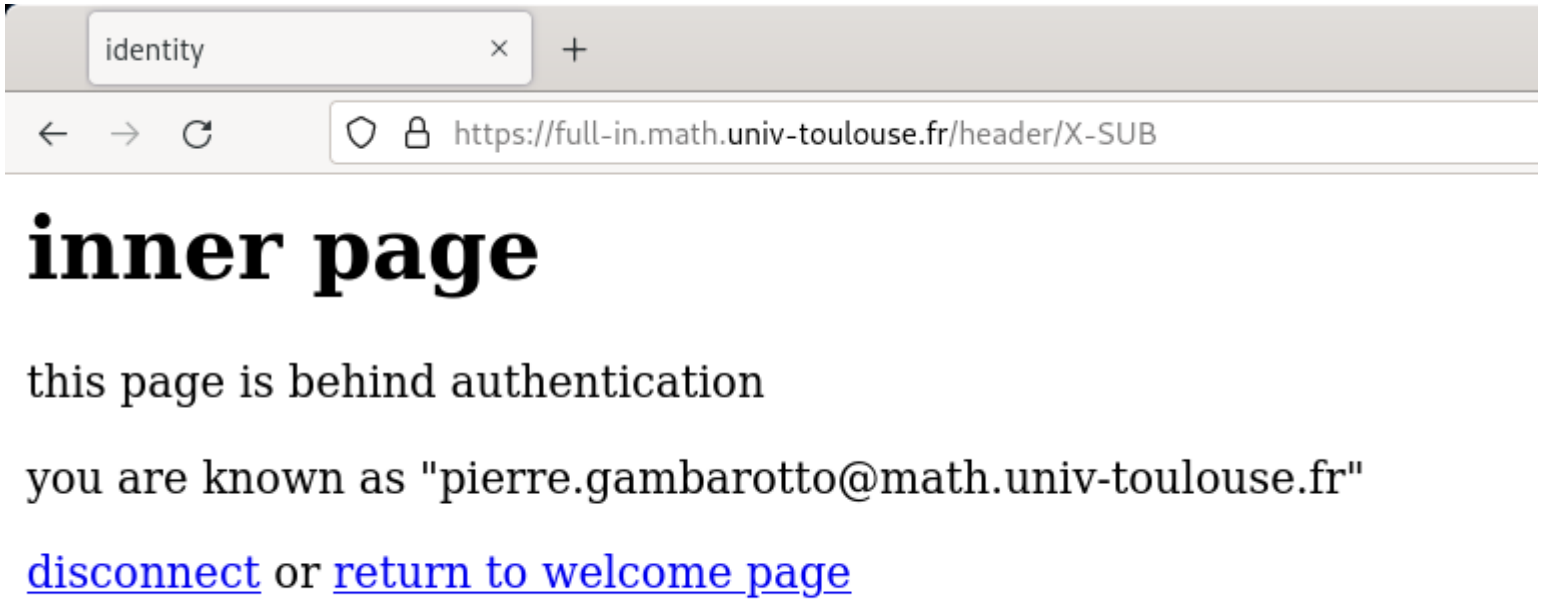
Mot de passe PLM oublié ?

FAQ CGU_PLM

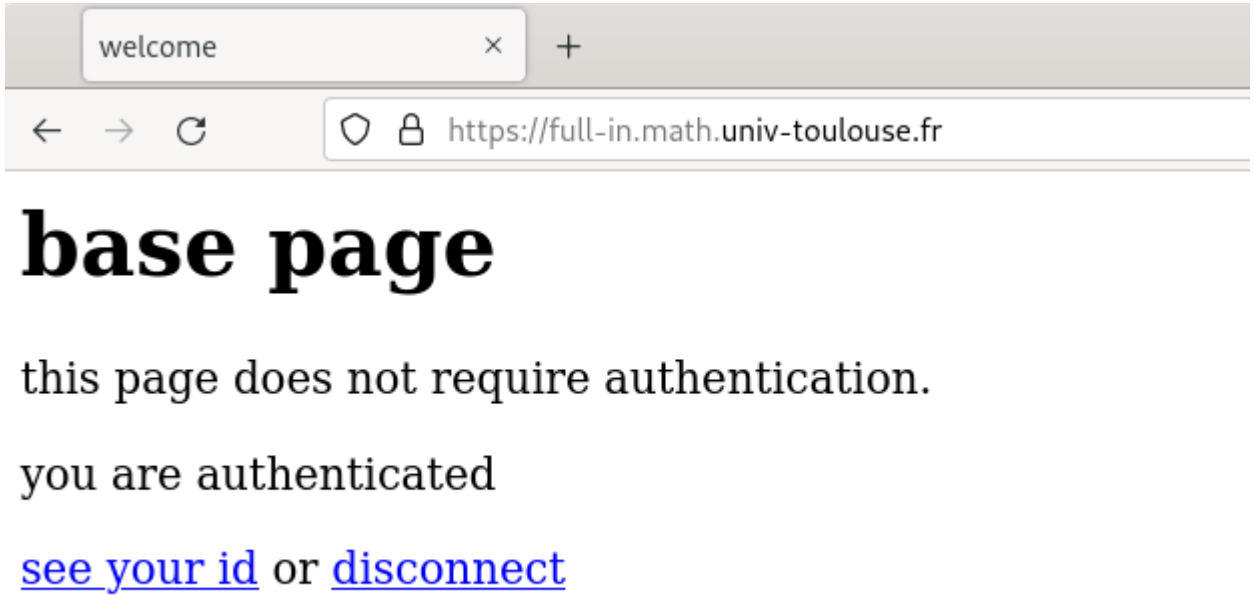
mathrice

cnrs INSMI

- après authentification : la page demandée initialement
- l'application a reçu des info sur la personne authentifiée



retour à la page de base, en mode authentifié



- déconnexion : https://full-in.math.univ-toulouse.fr/logout => retour à l'état initial

AU NIVEAU DÉVELOPPEUR

Pour adapter une application, il faut donc :

- page authentifiée ou pas :
  - lire un entête
  - logique en fonction de la présence de l'entête
- url pour la déconnexion : afficher le lien au bon endroit

Et c'est tout !

EXEMPLE EN HASKELL

```haskell
main :: IO ()
main = scotty 12345 $ do
  get "/" $ do                          -- base url : authenticated o
    mh <- header "X-SUB"                -- wether header "X-SUB" is p
    lucid $ template "welcome" $ do
     h1_ "base page"
     p_ "this page does not require authentication."
     case mh of
       Nothing -> a_ [ href_ "/header/X-SUB" ]   -- no header
         "authenticate then display subject identity"
       Just sub -> do                                -- header : a
         p_ "you are authenticated"
         a_ [ href_ "/header/X-Sub" ] "see your id"
         span_ " or "
         a_ [ href_ "/logout" ] "disconnect"
```

=> il suffit de récupérer un *header* HTTP

INFOS UTILISATEUR REMONTÉES

https://plm.math.cnrs.fr/oauth-client

```
extra > raw_info
```

STANDARD


```
{
  "sub": "pierre.gambarotto@math.univ-toulouse.fr",
  "eppn": "CNRS",
  "displayName": "Pierre GAMBAROTTO",
  "mailplm": "pierre.gambarotto@math.univ-toulouse.fr",
  "email": "pierre.gambarotto@math.univ-toulouse.fr",
  "name": "Pierre GAMBAROTTO",
  "preferred_username": "gamba",
  "family_name": "GAMBAROTTO",
  "given_name": "Pierre",
  "gender": "M."
}
```

SPÉCIFIQUE À LA PLM


```
"legacyplm": {
  "IDP": "CNRS",
  "UserName": "Pierre.Gambarotto@math.univ-toulouse.fr",
  "EPPN": "pierre.gambarotto@cnrs.fr",
  "MAIL": "Pierre.Gambarotto@math.univ-toulouse.fr",
  "MAILI": "pierre.gambarotto@math.univ-toulouse.fr",
  "MAILS": [
      "pierre.gambarotto@math.univ-toulouse.fr",
      "pierre.gambarotto@math.cnrs.fr"
  ],
  "emathMAILS": [
      "pierre.gambarotto@math.univ-toulouse.fr",
      "pierre.gambarotto@univ-tlse3.fr"
  ],
  "title": "M.",
  "MSC2010": [ ],
  "CN": "GAMBAROTTO Pierre",
  "SN": "GAMBAROTTO",
  "displayName": "Pierre GAMBAROTTO",
  "givenName": "Pierre"
}
```

```
"legacyplm": {
  "FROM": "emath",
  "STATUS": "plm",
  "USER": "gamba",
  "mailLAB": "pierre.gambarotto@math.univ-toulouse.fr",
  "OU": [
      "2802"
  ],
  "ADMIN": [
      "2802"
  ],
  "mailPLM": "Pierre.GAMBAROTTO@math.cnrs.fr",
  "LAB": [
      "138",
      "133"
  ]
}
```

- `"USER"` : l'identifiant plm, requiert `"STATUS": "plm"`
- `"FROM": "plm_emath"|"emath"`
- `"STATUS": "plm"|"insmi"`
- `"OU": [ "123", "456""]` : référence aux branches LDAP des labos

DÉPLOIEMENT AVEC ~~DEBIAN~~ NIXOS

- compilation nginx + modules (openidc, http, session, jwt, openssl) pas simple -> openresty

nix rend la reproduction *exacte* d'un environnement logiciel possible

nixos : idem au niveau OS

```
nixos-rebuild switch
# after, OS config is exactly as described in /etc/nixos/configura
```

NIXOS = NIX + COLLECTION DE MODULES SYSTEMD

- très bon module nginx
- let’s encrypt

CONFIGURATION PRINCIPALE

```
{ config, pkgs, ... }:
{
  imports = [
    <nixpkgs/nixos/modules/virtualisation/lxc-container.nix>
    ./network.nix
    ./nginx-reverse-proxy.nix
    # list of virtual hosts
    ./full-in.math.univ-toulouse.fr-host.nix
    ./www-thb2.math.univ-toulouse.fr-host.nix
    ./mint.math.univ-toulouse.fr-host.nix
  ];

  users.users.root.openssh.authorizedKeys.keys = [
    "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEpr3uamfBpDDtWA0G/vWNf/z
  ];

  environment.systemPackages = with pkgs; [
   vim
   minica
  ];
}
```

NGINX

Config de base :

```
{ config, lib, pkgs, ...}:
with lib;
{
  networking.firewall.allowedTCPPorts = [ 80 443 ];
  security.acme.email = "sysadmin@math.univ-toulouse.fr";
  security.acme.acceptTerms = true;
  services.nginx = {
    enable = true;
    logError = "stderr debug";
    enableReload = true; # systemd reload (instead of restart) whe
    recommendedProxySettings = true;
    recommendedTlsSettings = true;
    recommendedGzipSettings = true;
    resolver.addresses = [ "130.120.36.77" ];
}
```

NGINX, LES AJOUTS POUR QUE LUA GÈRE OIDC

```
{   package = pkgs.openresty; # ngnix version by openresty.org
    appendHttpConfig = let
      extraPureLuaPackages = with pkgs.luajitPackages; [
        lua-resty-openidc lua-resty-http
        lua-resty-session lua-resty-jwt lua-resty-openssl
      ];
      luaPath = pkg: "${pkg}/share/lua/5.1/?.lua";
      makeLuaPath = lib.concatMapStringsSep ";" luaPath;
    in ''
      lua_package_path '${makeLuaPath extraPureLuaPackages};;';
      lua_ssl_trusted_certificate /etc/ssl/certs/ca-certificates.c
      lua_ssl_verify_depth 5;

      # cache for OIDC discovery metadata
      lua_shared_dict discovery 1m;
      # cache for JWKs.
      lua_shared_dict jwks 1m;
    '';
  };
  systemd.services.nginx.serviceConfig = {
```

```
    MemoryDenyWriteExecute = lib.mkForce false;  # openresty; now
  };
}
```

VIRTUALHOST, BASE

Pas d'authentification gérée, on ne fait que la terminaison SSL :

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

let
    proxy-hostname = "mint.math.univ-toulouse.fr";
    proxy-uri = "https://${proxy-hostname}";
    backend-uri = "http://web-mint-preprod.math.univ-toulouse.fr:8
in
{
  services.nginx.virtualHosts."${proxy-hostname}" =  {
    enableACME = true;
    forceSSL = true;

    locations."/" = {
      proxyPass = "${backend-uri}";
      proxyWebsockets = true; # needed if you need to use WebSocke
    };
  };
}
```

VIRTUALHOST : AUTHENTIFICATION PARTOUT

Idem, plus :

```
let
    luaAccess = /root/openid.lua;
in
{
  services.nginx.virtualHosts."${proxy-hostname}" =  {
    enableACME = true;
    forceSSL = true;

    locations."/" = {
      proxyPass = "${backend-uri}";
      proxyWebsockets = true; # needed if you need to use WebSocke
      extraConfig = ''
        access_by_lua_file "${luaAccess}";
      '';
    };
  };
}
```

Les requetes sont en plus évaluées par le code lua.

PARTIE PUBLIQUE, PARTIE PRIVÉE

```
{ ... }:
let
    luaAccess = /root/openid.lua;
in
{
  services.nginx.virtualHosts."test-in.math.univ-toulouse.fr" =  {
    enableACME = false;
    forceSSL = true;
    sslCertificate = /root/star.math.univ-toulouse.fr/star.math.un
    sslCertificateKey = /root/star.math.univ-toulouse.fr/star.math

    locations."/auth" = {
      proxyPass = "http://130.120.36.142:12345/test";
      proxyWebsockets = true; # needed if you need to use WebSocke
      extraConfig = ''
        access_by_lua_file "${luaAccess}";
      '';
    };
    locations."/" = {
      proxyPass = "http://130.120.36.142:12345";
      proxyWebsockets = true; # needed if you need to use WebSocke
    };
```

```
    };
}
```

CONFIG NGINX GÉNÉRÉE

Redirection 80 -> 443

```
server {
        listen 0.0.0.0:80 ;
        listen [::]:80 ;
        server_name test-in.math.univ-toulouse.fr ;
        location / {
                return 301 https://$host$request_uri;
        }
}
```

```
server {
        listen 0.0.0.0:443 ssl http2 ;
        listen [::]:443 ssl http2 ;
        server_name test-in.math.univ-toulouse.fr ;
        ssl_certificate /nix/store/65pmm728jri4hp93js1v4gfhbqxj5sz
        ssl_certificate_key /nix/store/csz6wcafvgb0kamra0zhzqx2fyj
        location / {
                proxy_pass http://130.120.36.142:12345;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection $connection_upgrade;
                include /nix/store/c4sr5qbsawabkxgpqsh7aas6khjbvv3
        }
        location /auth {
                proxy_pass http://130.120.36.142:12345/test;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection $connection_upgrade;
                access_by_lua_file "/nix/store/kzplh5ra4scbsvzfkds
                include /nix/store/c4sr5qbsawabkxgpqsh7aas6khjbvv3
        }
}
```

EXEMPLES DE CONFIGURATION EN LUA

Un fichier lua par site virtuel :

- les informations d’enregistrement de l’application
- config oidc (server, scopes)
- la définition de l’url de logout
- logique pour les pages nécessitant l’auth
- logique pour les pages mixtes

BASIQUE : TOUT EST PROTÉGÉ

```
local opts = {
  redirect_uri = "https://test-in.math.univ-toulouse.fr/auth/callb
  discovery = "https://plm.math.cnrs.fr/sp/.well-known/openid-conf
  client_id = "edc62……65e1d0",
  client_secret = "4fa6……98a3f1",
  scope = "openid profile legacyplm",
  logout_path = "/auth/logout",
  redirect_after_logout_uri = "https://test-in.math.univ-toulouse.
}
local res, err = require("resty.openidc").authenticate(opts)
ngx.req.set_header("X-SUB", res.id_token.sub)
```

SIMPLE : PAGE D'ACCUEIL MIXTE

La config

```
 1: local proxy_hostname = "full-in.math.univ-toulouse.fr"
 2: local proxy_uri = "https://"..proxy_hostname
 3: local proxy_callback_uri = "/callback"
 4: local proxy_logout_uri = proxy_uri.."/logout"
 5: local proxy_public_uri = "/"
 6: local post_logout_uri = "/"
 7: local opts = {
 8:   redirect_uri = proxy_uri .. proxy_callback_uri,
 9:   discovery = "https://plm.math.cnrs.fr/sp/.well-known/openid-
10:   client_id = "edc……c65e1d0",
11:   client_secret = "4fa6……1698a3f1",
12:   scope = "openid profile legacyplm",
13:   logout_path = "/logout",
14:   redirect_after_logout_uri = "https://plm.math.cnrs.fr/sp/use
15: }
```

la logique

```
16: if ngx.var.request_uri == "/" then
17:   print("public uri, not authenticated")
18:   -- check if already authenticated
19:   local res, err = require("resty.openidc").authenticate(opts,
20:   if not err and (res and res.id_token) then
21:     print("==== authenticated =====")
22:     ngx.req.set_header("X-SUB", res.id_token.sub)
23:     ngx.req.set_header("X-EMAIL", res.user.email)
24:     ngx.req.set_header("X-DISPLAYNAME", res.user.displayName)
25:   end
26: else
27:   print("private uri")
28:   local res, err = require("resty.openidc").authenticate(opts)
29:   ngx.req.set_header("X-SUB", res.id_token.sub)
30:   ngx.req.set_header("X-EMAIL", res.user.email)
31:   ngx.req.set_header("X-DISPLAYNAME", res.user.displayName)
32: end
```

PROBLÈMES RENCONTRÉS

- lua et debug/log
- print debug dans les log, accessible dans les logs du service.

```
function dump(o)
    if type(o) == 'table' then
        local s = '{ '
        for k,v in pairs(o) do
                if type(k) ~= 'number' then k = '"'..k..'"' e
                s = s .. '['..k..'] = ' .. dump(v) .. ','
        end
        return s .. '} '
    else
        return tostring(o)
    end
end
```

DÉCO ET URL EN SUIVANT

le code existant du server oidc utilisait l'entete `Referer`

Firefox 87 (et chrome équivalent): valeur de `Referer` purgée dès que l'on change de domaine

Mathrice :

- mattermost avec Philippe
- issue sur gitlab, en proposant une solution
- intégration en suivant

Ajout d’un paramètre de requete pour gérer l’url de redirection post logout

```ruby
class ApplicationController < ActionController::Base
# ...
  def after_sign_out_path_for(resource_or_scope)
    cookies.delete(:'plm.session')
    cookies.delete(:ezproxy)
    session.clear
    return_to = request.params[:return_to] || request.referrer # e
    return_to ? "https://plm.math.cnrs.fr/Shibboleth.sso/Logout?re
  end
```

BILAN

- POC ok
- le serveur oidc ne voit que le reverse proxy : une seule déclaration d’application nécessaire, pas une par virtualhost
- nixos : impressionné

POUR LA SUITE

CÔTÉ INFRA : OPENSHIFT !

- générer un conteneur à partir d'une expression nix
- on perd l'enrobage systemd/nixos
- un conteneur ~ un service, découpage différent

CÔTÉ LOGIQUE

- lua json pour récupérer OU et vérifier le labo d'appartenance
- déléguer l'enregistrement à un utiilsateur

DÉPLOIEMENT

- nix pour générer le code lua pour des configurations types