entr'ouvert

publik

le connecteur citoyen

Architecture technique

Documentation

Table des matières

Présentation générale ... 3

Infrastructure de base requise ... 4
1 - Machines nécessaires ... 4
2 - Machine PostgreSQL ... 4
3 - Composants logiciels sous-jacents ... 4
3.1 - Source des logiciels (dépôts APT) ... 4
4 - Flux réseau à ouvrir ... 5
5 - Certificats X509 ... 5
6 - MCO par Entr'ouvert ... 5

Architecture d'une brique logicielle ... 7
1 - Utilisation de cadriciels (*frameworks*) ... 7
2 - Schéma ... 8
3 - Cas particulier de la brique w.c.s. ... 8

Dialogues entre briques ... 10
1 - API Publik ... 10
1.1 - Description générale ... 10
1.2 - Exemple d'un dialogue combo / w.c.s. ... 12
2 - Messagerie pour provisionning ... 13

Infrastructure d'hébergement SaaS ... 14

Autres infrastructures possibles ... 16
1 - Infrastructure PoC / test ... 16
2 - Infrastructure légère et « élastique » ... 16
3 - Infrastructure pour hébergement sur site ... 17

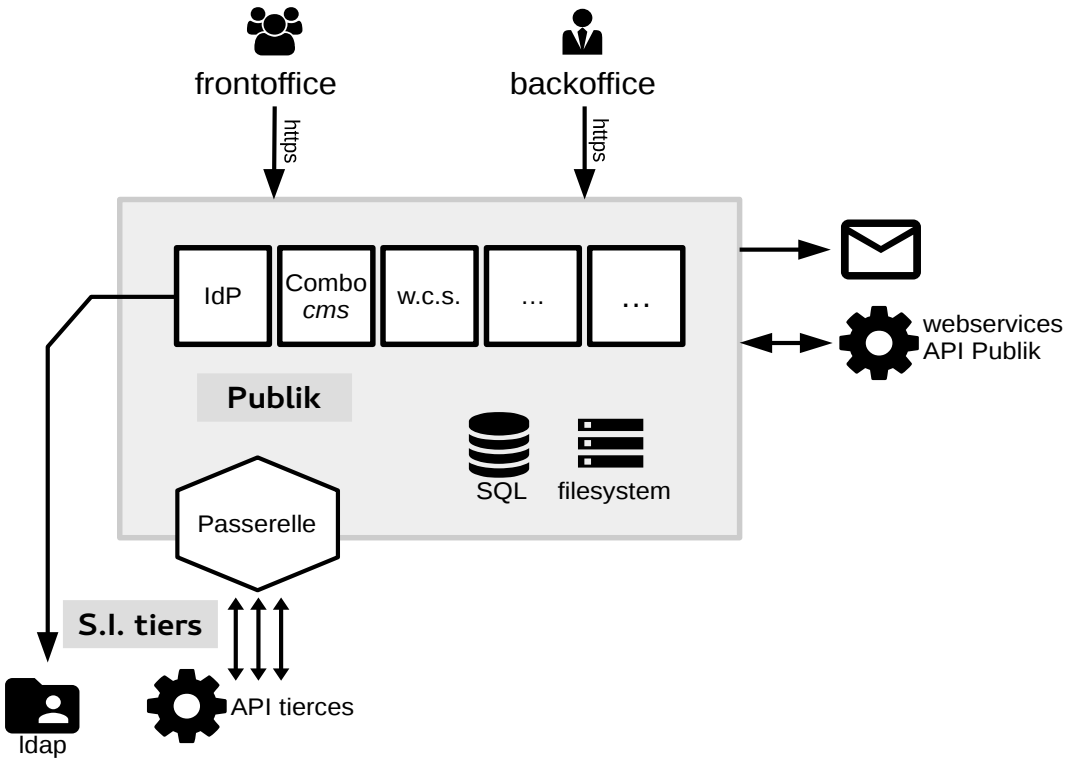
Accès à un S.I. tiers ... 18
1 - Protection des accès webservices ... 18
2 - Protection de l'accès LDAP : X509 ... 18
3 - Ajout de protections sur webservices existants ... 19
3.1 - Ajout d'un *reverse-proxy* ... 19
3.2 - Ajout d'une instance passerelle « locale » ... 19
3.3 - Accès via VPN ... 20

Ce document présente l'architecture technique d'un système Publik nécessaire pour garantir un fonctionnement sécurisé et permanent. Il se concentre sur les briques logiciels « actives » et ne détaille pas les modules nécessaires classiques à tout système d'information : supervision, logs, sauvegarde, technique de virtualisation, orchestration des configuration, architecture des machines, etc.

Présentation générale

Publik est un ensemble de composants : on parle des « briques » de la solution. Elles sont représentées ci-dessous sous la forme de petits carrés. Elles sont accessibles par les humains via le web (HTTPS) au travers d'interfaces « frontoffice » ou « backoffice ».

Le composant « passerelle » est particulier, il assure la connexion avec des systèmes tiers en traduisant les webservices internes de Publik en protocoles et formats des logiciels cibles.



Liste des briques disponibles :

- authentic : gestion des identités, IdP (*identity provider*)
- combo : CMS pour portails usager et agent (porte d'entrée de Publik)
- w.c.s. : formulaires et workflows
- passerelle : connecteurs vers systèmes tiers
- fargo : porte-documents
- corbo : diffusion de messages
- chrono : prise de rendez-vous
- welco : interface de saisie (multi-canal)
- hobo : système de déploiement et de provisionning

Un système Publik installé ne comporte pas obligatoirement toutes les briques.

Infrastructure de base requise

1 - Machines nécessaires

Idéalement, pour une meilleure isolation et une meilleure sécurité, chaque brique dispose de son propre serveur. Cependant il est possible de varier les installations, jusqu'à tout installer sur une seule machine (par exemple lors de tests, de PoC, de développements).

Machine recommandée pour opérer une brique :

- processeur architecture x86-64 double cœur
- mémoire vivre 4Go
- volume disque fonction de l'usage prévu, typiquement 10Go (minimum 1Go pour un test)

Les briques w.c.s. (formulaires) et fargo (porte-documents) doivent disposer d'un volume plus important pour gérer les documents des usagers. Ceux-ci sont stockés sur le système de fichier, son volume est donc à évaluer en fonction de l'usage prévu.

Il est fortement conseillé de disposer d'un pool de machines virtuelles dont la puissance et les volumes disques peuvent être modifiés *à chaud*. La technologie de containers apporte cette souplesse avec un minimum de perte de puissance.

2 - Machine PostgreSQL

Chaque brique utilise une ou plusieurs bases de données PostgreSQL, il faut donc disposer d'une machine pour cela. Il est courant d'utiliser une installation à deux machines identiques en master/slave, avec les mêmes recommandations que ci-dessus.

Publik peut utiliser un système PostgreSQL existant, le cas échéant.

3 - Composants logiciels sous-jacents

Publik est un logiciel développé en Python 2.7, sur le cadriciel (*framework*) Django 1.8. Il est prévu pour fonctionner sur un système d'exploitation GNU/Linux et sa distribution officielle (par paquets) cible Debian GNU/Linux en version stable ou oldstable.

Le frontal web recommandé est nginx, bien que Publik puisse fonctionner avec Apache et d'autres serveurs HTTP. La liaison entre les applicatifs Python et le frontal web est assurée par gunicorn (évolution en cours vers uwgsi).

Les différents composants (briques) de Publik échangent des messages AMQP via RabbitMQ.

Publik nécessite PostgreSQL en version 9 (> 9.4 recommandée) sur lequel chaque brique disposera de sa base de données propre.

3.1 - Source des logiciels (dépôts APT)

Les différents composants logiciels utilisés par Publik proviennent, par ordre de

préférence :

- de la distribution Debian GNU/Linux stable (ou oldstable)
- des *backports* officiels Debian, disposant du suivi de sécurité par l'équipe Debian
- des paquets Debian fournis et maintenus par les projets *upstream*
- de paquets Debian maintenus par Entr'ouvert qui en assure le suivi de sécurité

4 - Flux réseau à ouvrir

Publik est un système web : ses composants doivent être accessibles en HTTPS (443/tcp). Le S final est important : Publik n'est pas prévu pour fonctionner en HTTP.

Les différentes briques de Publik communiquent entre elles également en HTTPS, elles doivent donc disposer d'un accès DNS (53/udp et 53/tcp) et HTTPS.

Le système de provisonning (déploiement, configuration et diffusion des utilisateurs et des rôles) utilise AMQP (5671/tcp).

La connexion avec PostgreSQL utilise 5432/tcp.

La plupart des accès à des logiciels tiers se fait via webservice, généralement en HTTPS (443/tcp). Il peut également y avoir connexion à des annuaires LDAP (389/tcp).

La solution Publik n'a pas encore été validée sur IPv6.

Pour le support et la maintenance du systèmes, des accès SSH (22/tcp) sont nécessaires, voir ci-dessous.

5 - Certificats X509

La diffusion HTTPS étant obligatoire, il est nécessaire de disposer de certificats valides pour chaque brique déployée ; chacune des briques utilisant un nom de serveur distinct. En général un certificat étoile (*wildcard* *.example.net) couvre toutes les briques.

6 - MCO par Entr'ouvert

Pour qu'ils puissent assurer le maintien en conditions opérationnelles (MCO) de la partie logicielle Publik, les travailleurs d'Entr'ouvert doivent :

- avoir un accès SSH aux machines le plus direct possible (sans nécessité de passer par un VPN, qui plus est s'il est propriétaire ou exotique). Entr'ouvert peut indiquer une liste d'adresses IPv4 source ;
- disposer d'un accès administrateur (root) sur les machines, via *su* ou *sudo*.

Cet accès n'est pas demandé sur les machines d'infrastructure « non logicielle » telle que le serveur de base de données PostgreSQL, le serveur de log, les backups, etc. La maintenance de ces systèmes est laissée aux opérateurs habituels du site ; sauf contrat spécifique avec Entr'ouvert.

Si l'accès web à la solution est fermé, par exemple dans le cas d'un PoC ou d'une utilisation interne, alors cet accès doit être possible pour Entr'ouvert (là encore, accès direct sans VPN si possible).

Si les permissions *root* sont impossibles, il est au moins nécessaire, pour des raisons de

support, qu'Entr'ouvert accéde facilement aux logs des machines voire à son système de supervision, et ce en temps réel. Il doit par ailleurs exister une procédure de passage root en cas d'urgence. Enfin, si Entr'ouvert n'a pas d'accès *root* à la machine, cela signifie que la supervision et les mises à jour régulières (au moins quotidiennes) sont opérées par l'hébergeur ; condition sans laquelle Entr'ouvert ne peut pas garantir une sécurité maximale du système.

Enfin, Entr’ouvert ne peut assurer un MCO efficace que sur des machines Debian GNU/Linux maintenues à jour – cette maintenance peut même être assurée par Entr’ouvert. Pour tout autre système d’exploitation, un contrat spécifique doit être prévu.

Architecture d'une brique logicielle

1 - Utilisation de cadriciels (*frameworks*)

Chaque brique logicielle est une application Python/Django – à l'exception de la brique w.c.s. qui utilise le cadriciel Quixote.

L'utilisation d'un *framework* permet de disposer d'un ensemble de composants afin de développer plus rapidement, mais aussi de manière plus homogène, et surtout en assurant à tout moment une excellent sécurité de l'application. C'est en effet le framework qui :

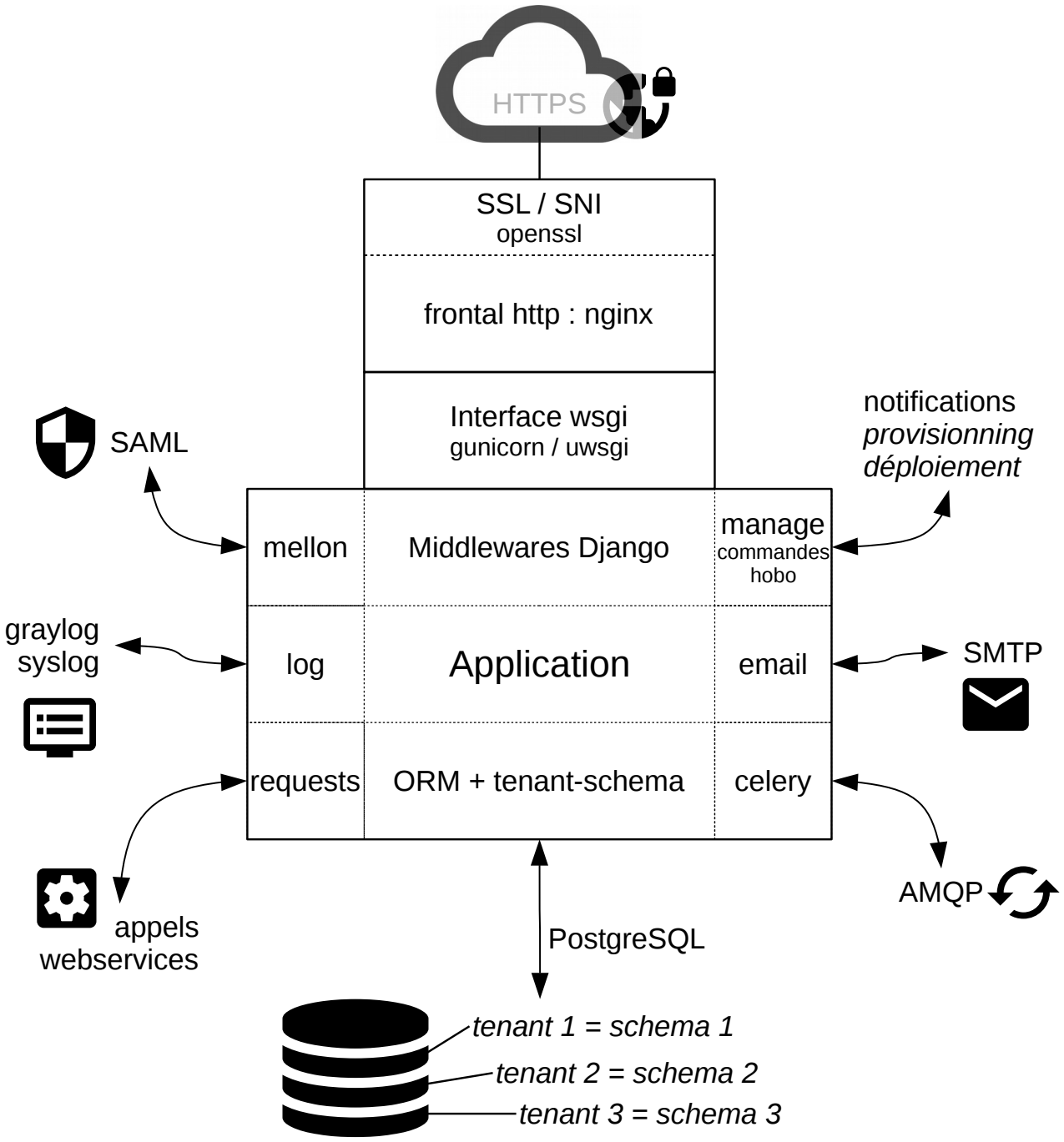
- reçoit les données, les interprète et les valide avant de les envoyer à l'application ;
- permet de contrôler l'envoi des données aux bases de données par l'application (pas de requêtes SQL directes) ;
- sécurise les sorties de l'application (HTML) en imposant un contrôle fort sur les données affichable.

Dans Publik, en plus de l'utilisation de toutes les possibilités de Django (ou Quixote), d'autres protections sont mises en place :

- isolation des composants (chaque composant est une brique logicielle indépendante) ;
- chaque brique dispose de sa propre base de données, complètement isolée des autres (chaque base peut même être hébergé sur un serveur propre) ;
- chaque brique peut gérer plusieurs sites (mode multi-tenant), dans ce cas chaque site dispose d'un « tenant » dans la base de données sous forme d'un schéma PostgreSQL : chaque site est donc indépendant et isolé ;
- utilisation du front-end nginx pour diffuser tous les éléments statiques des applications ;
- connexion de l'application via le protocole wsgi pour un premier filtrage des requêtes (les requêtes invalides sont rapidement éliminées).

D'une façon générale, Publik utilise au maximum des composants éprouvés : le code des applications se concentre uniquement sur la logique de celle-ci. Il s'agit de suivre les principes DRY (Don't Repeat Yourself) et KISS (Keep It Stupid Simple) afin de mieux sécuriser l'application : la sobriété recherchée par Publik est aussi présente dans le code du logiciel.

2 - Schéma



Le schéma ci-dessus montre que l'application n'est pas en contact « direct » avec l'extérieur. Elle utilise toujours des composants logiciels soit éprouvés (celery, requests, tenant-schemas) soit communs à toutes les briques (hobo, mellon).

3 - Cas particulier de la brique w.c.s.

De conception plus ancienne, la brique w.c.s. utilise un autre framework, Quixote. Il apporte les mêmes avantages en terme de sécurité que Django tout en offrant une plus

grande souplesse au niveau des objets gérés. Cette souplesse est le cœur de l'application w.c.s. : création de formulaires et de formulaires « libres », actions de workflow souple, utilisation importante des possibilités objet du langage Python (stockage des objets natifs, via « cPickle »).

La gestion SQL des données de w.c.s. est également spécifique, avec la construction de vues nécessaires aux optimisations pour les statistiques et « vues globales » de w.c.s.

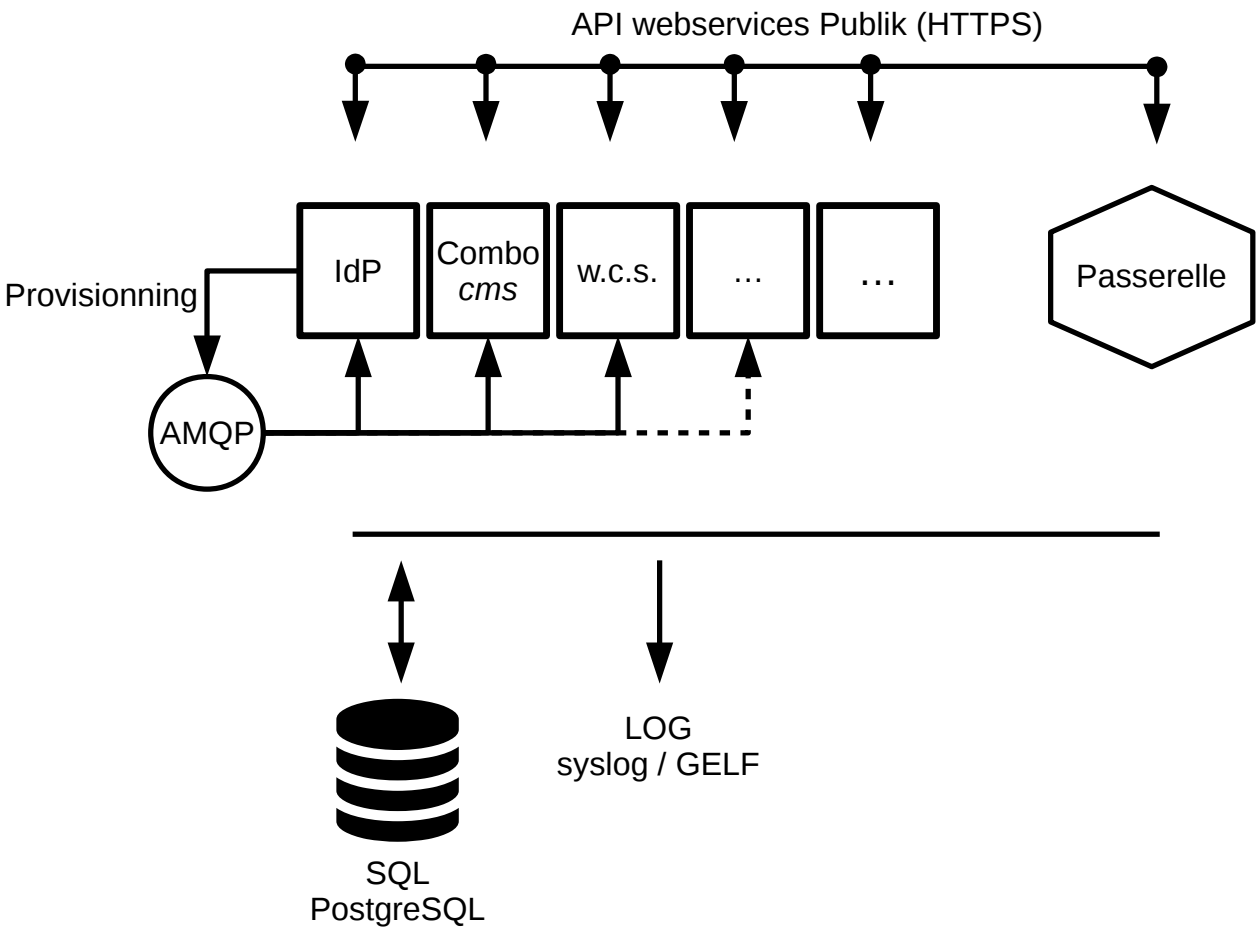
Un portage vers le cadriciel Django est actuellement en cours pour, dans un premier temps, profiter des middleware et du système de templates de Django. Pour le reste, w.c.s. utilise déjà des composants identiques aux autres applications de Publik, et le schéma précédent peut-être repris en remplaçant « Django » par « Quixote ».

Dialogues entre briques

Les briques de Publik dialoguent via deux canaux :

- webservices (HTTPS JSON) pour ce qui concerne l'échange de données
- messages (AMQP) pour ce qui concerne la gestion des utilisateurs et des rôles, *i.e.* le provisionning au travers des différents composants

Par ailleurs SAML est utilisé pour ce qui concerne le WebSSO (le dialogue se déroule via le navigateur de l'usager).



1 - API Publik

1.1 - Description générale

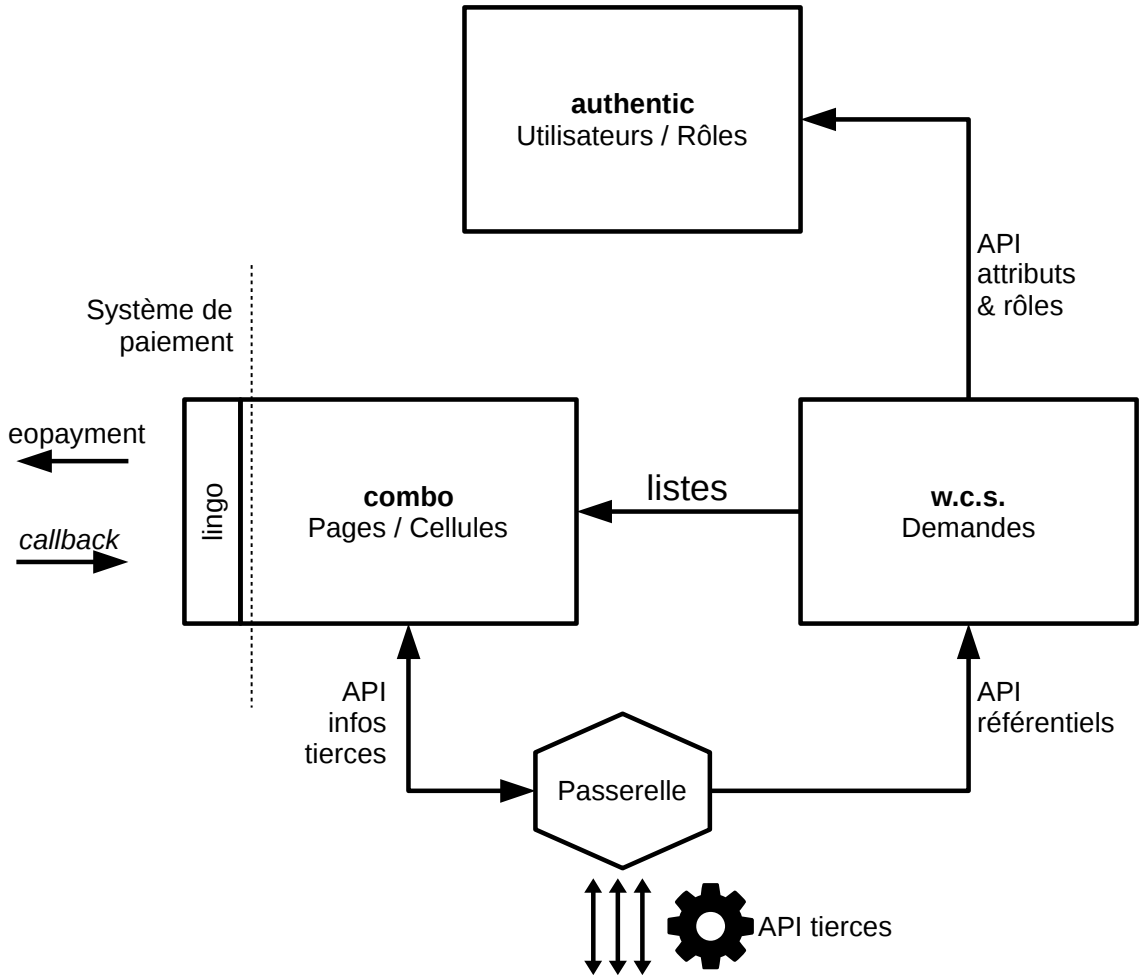
L'API des webservices Publik permet aux briques d'échanger des données. La plupart ont trait à l'échange de données autour d'un usager.

Fournisseurs d'API :

- w.c.s. est la brique qui propose le plus de webservices autour des demandes d'un usager (http://doc.entrouvert.org/wcs/dev/#api) ;
- passerelle met en place des connecteurs donc la plupart ont pour objectif de remonter les informations d'un système tiers concernant un usager ;
- passerelle est aussi utilisé pour remonter des informations de type « référentiels » depuis des systèmes tiers ;
- authentic propose des API permettant la gestion des rôles, des usagers (attributs et rôles).

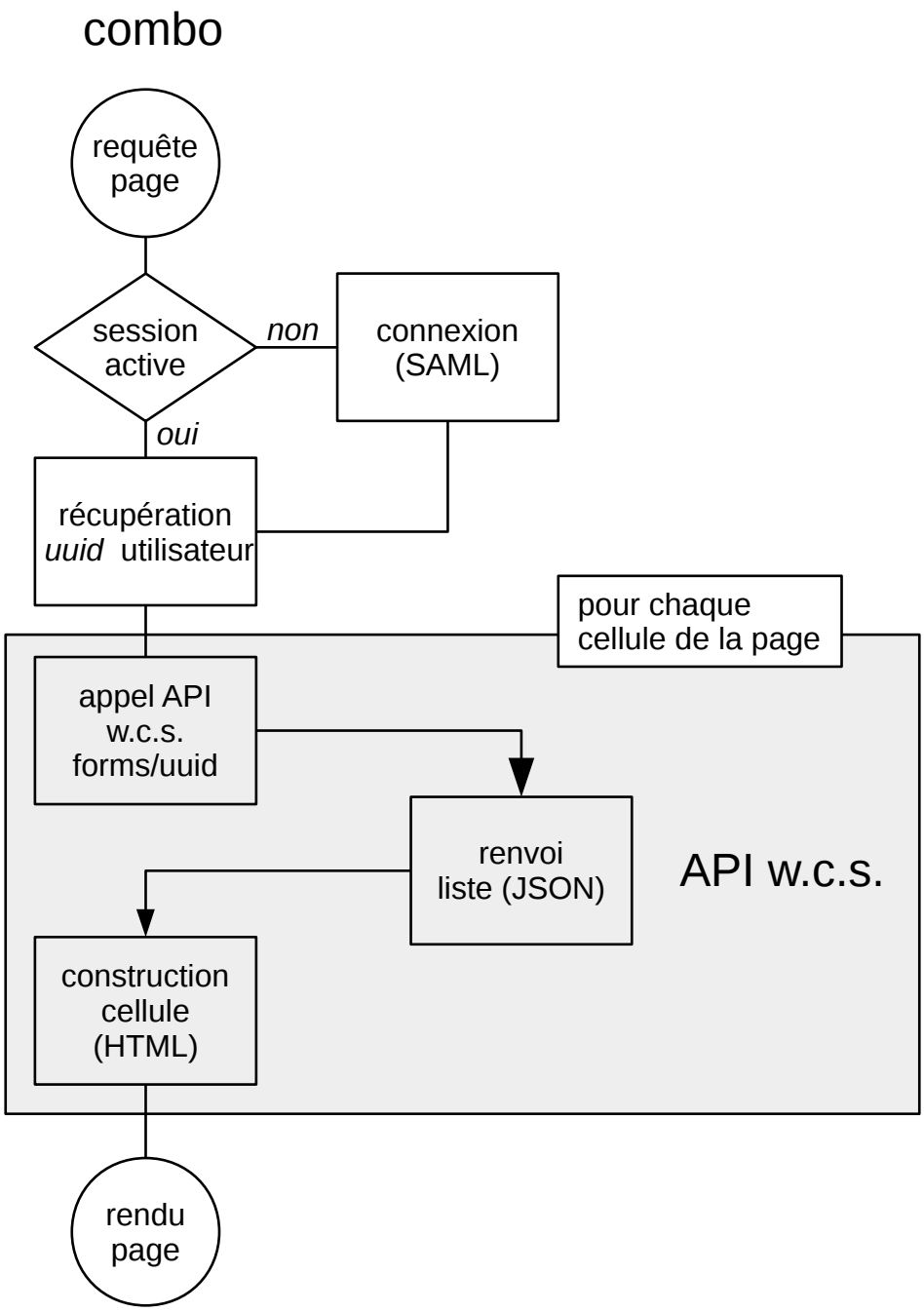
Consommateurs d'API :

- combo est un consommateur de ces API afin de présenter à l'usager l'ensemble des données que Publik connaît le concernant ;
- w.c.s. utilise les API de passerelle pour présenter des référentiels dans les formulaires ;
- w.c.s. utilise les API d'Authentic dans ses actions de workflow afin de pouvoir gérer les rôles d'un usager qui a fait une certaine demande ;
- w.c.s. peut également faire appel à tout webservice de Passerelle afin d'intervenir sur un système tiers, toujours lors d'action de workflow (typiquement pour injecter des données dans un système tiers).



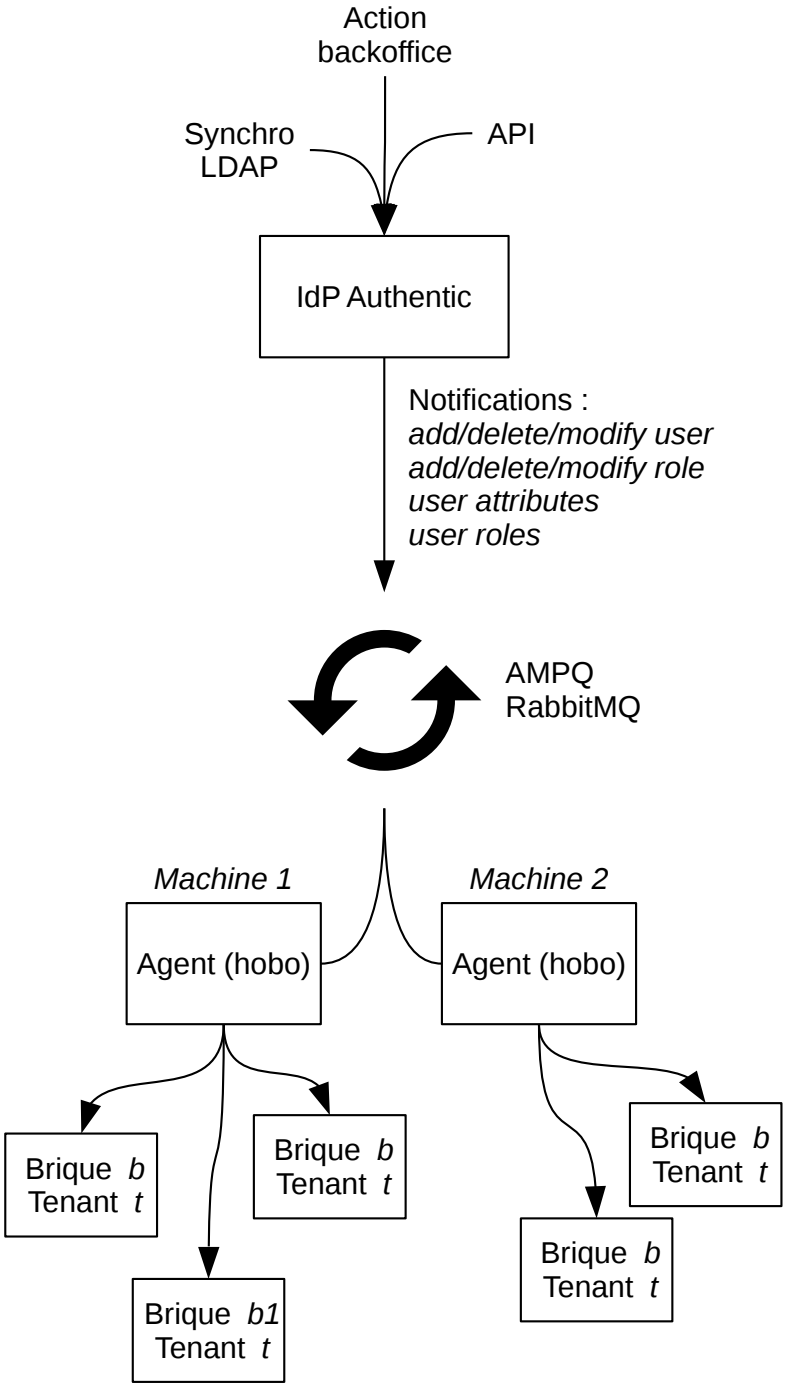
1.2 - Exemple d'un dialogue combo / w.c.s.

Objectif : lorsque l'usager se connecte sur son portail usager, une cellule de la page affiche la liste de ses demandes en cours. Pour cela, combo va faire appel à différents webservice de w.c.s. qui lui retourneront les informations nécessaires à afficher dans les cellules de la page.



2 - Messagerie pour provisionning

Le système de provisionning des utilisateurs et des rôles opère via un diffuseur de message AMQP (RabbitMQ). Lorsqu'une modification est effectuée sur la brique authentic, celle-ci diffuse un message à l'attention de toutes les instances des autres briques concernées. Le message est reçu par tous les agents de toutes les machines de l'infrastructure, et est diffusé aux briques concernées, qui se chargent de modifier les utilisateurs des instances cibles.



Infrastructure d'hébergement SaaS

L'infrastructure d'hébergement proposée par Entr'ouvert utilise les techniques suivantes :

- des machines physiques au matériel redondant (alimentation double, disques durs RAID 1) avec de très importantes capacités CPU (au moins 16 coeurs) et mémoire vive (au moins 128Go) : ces machines sont capables de supporter sans délai des montées en charges brusques ;
- ces machines disposent de disques ultra-rapides (SSD) pour fournir des espaces disques locaux véloces si nécessaire (espace système, espaces temporaires) ; ainsi que de disques rapides (SAS ou SATA) pour des espaces de stockages locaux importants (backups locaux, fichiers transitoires) ;
- des machines virtuelles basées sur un système de container (OpenVZ, en migration vers lxc) permettant de dimensionner la puissance CPU ou la mémoire vive de chaque machine *à chaud* ;
- des espaces disques obtenus par NFSv3 depuis un SAN, permettant un redimensionnement à chaud des volumes disque ; par ailleurs le SAN utilise la technologie ZFS avec des *snapshots* horaire, quotidien et hebdomadaires ;
- les machines virtuelles utilisent des IP virtuelles (« failover ») qui peuvent être déplacées en quelques minutes vers une autre machine physique, et ainsi ne pas nécessiter de modification DNS en cas de déplacement des VM.

En terme de sécurité de données :

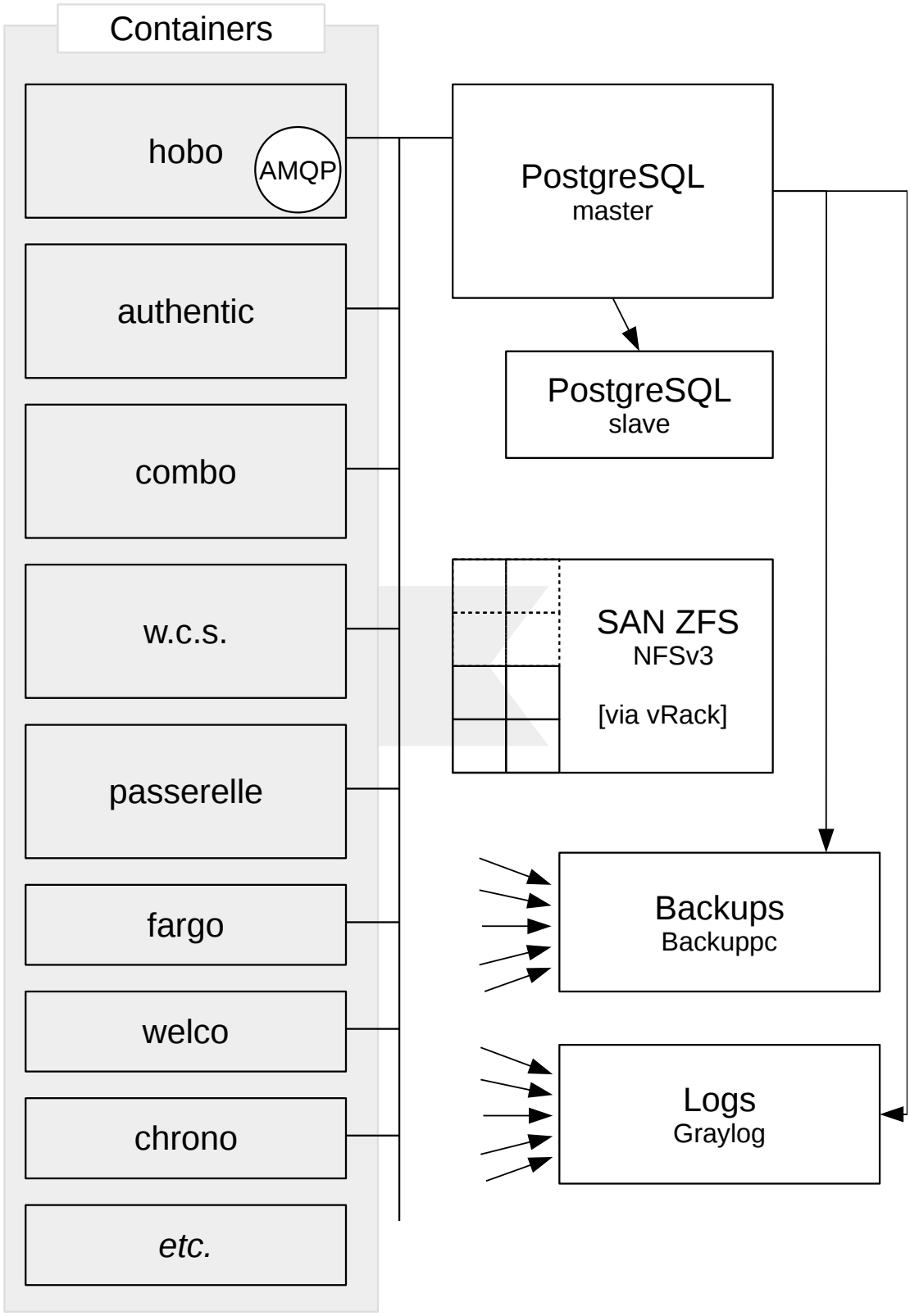
- En plus des snapshots ZFS, des backups incrémentaux (quotidiens) et complets (hebdomadaires) sont effectués sur une machine isolée hors site ;
- Le système de restauration est régulièrement vérifié ;
- La duplication d'une machine ou son transport vers un autre hôte physique peut être effectuée en quelques minutes (modulo propagation des IP fail-over) ;
- Le serveur PostgreSQL est une machine à disque RAID-5, sauvegardée quotidiennement. Elle est secondée d'une machine « slave » capable de prendre le relais en cas de défaillance.

En terme de sécurité réseau :

- Un pare-feu est présent sur chaque machine physique, en entrée comme en sortie. En dehors des accès HTTPS aux briques applicatives, tous les flux sont filtrés par défaut et ouvert uniquement au cas par cas ;
- Une surveillance « fail-to-ban » coupe les IP qui tentent des connexions non autorisées ;
- L'hébergeur des machines physiques assure par ailleurs une protection anti-DDOS.

Un serveur de log externe regroupe tous les événements de toutes les installations.

Une supervision générale est assurée par des agents SNMP et NRPE qui remontent les informations sur une plateforme Nagios et alertent les travailleurs d'Entr'ouvert en temps réel.


Containers
hobo
AMQP
authentic
combo
w.c.s.
passerelle
fargo
welco
chrono
etc.
PostgreSQL
master
PostgreSQL
slave
SAN ZFS
NFSv3
[via vRack]
Backups
Backuppc
Logs
Graylog

Autres infrastructures possibles

1 - Infrastructure PoC / test

Dans le cadre d'une installation de test ou PoC (preuve de concept) il est habituel d'installer Publik sur une seule machine. Dans ce cas, toutes les briques partagent :

- un seul frontal ngnix
- un seul serveur de base de donnée
- des espaces disques locaux (`/var/lib/<brique>`)
- un système de log (en général le *syslog* système)

Dans ce cadre, Publik peut être installé sur une machine aujourd'hui habituelle :

- processeur x86-64 double cœur
- mémoire vive 2Go
- espace disque 10Go, voire moins

Cependant et pour rappel, Publik nécessitant un fonctionnement HTTPS, il reste nécessaire de disposer :

- d'un enregistrement DNS par brique instanciée ;
- d'un certificat x509 valable pour chaque brique, généralement un wildcard sur le domaine choisi.

2 - Infrastructure légère et « élastique »

Entre une infrastructure SaaS telle que celle gérée par Entr'ouvert capable de répondre à des dizaines d'installations, et une infrastructure minimale telle que celle décrite pour un PoC, toutes les combinaisons sont possibles.

La partie la plus délicate à gérer est souvent la partie x509, et parfois la partie DNS quand il est question de pouvoir créer de nouvelles instances automatiquement dans le cadre d'un ensemble de communes (syndicat, organisme, métropole ou agglomération désirant diffuser la solution Publik à ses « membres »).

Pour rendre l'installation « élastique », c'est-à-dire capable de s'adapter au futur, il est conseillé :

- de virtualiser toutes les machines ;
- d'utiliser une technique de virtualisation permettant des modifications CPU, RAM et disque faciles, idéalement à chaud ;
- de disposer d'un serveur PostgreSQL central ;
- d'avoir accès à un SAN pour le stockage des données ;
- d'avoir toute liberté sur la partie DNS, éventuellement via des CNAME ou une délégation de zone ;
- de disposer d'un certificat x509 wildcard pour chaque domaine à gérer.

Une fois l'installation effectuée, il est souvent assez simple de permettre le déplacement d'une brique vers une autre machine : copie des configurations et des données, modification DNS, l'opération correctement préparée est sans risque et ne provoque pas

de coupure de plus de 10 minutes.

3 - Infrastructure pour hébergement sur site

Un hébergement sur site est de type « élastique » (cf *supra*), voici les recommandations d'usage pour l'initialisation :

- machine « IdP » pour authentic (utilisable par d'autres systèmes que Publik)
- machine « applications » : hobo + combo + w.c.s. + fargo + passerelle
- machine PostgreSQL
- autres briques sur une autre machine ou sur la machine « applications »
- backups et redondance (*fail-over*) assurés par ailleurs

Pour une collectivité avec plusieurs déploiements prévus à l'initialisation, la machine « applications » peut être scindée :

- machine hobo + combo + passerelle
- machine w.c.s. + fargo (porte-documents)
- autres briques sur une machine tierce

Toutes les machines sont virtuelles et modifiables rapidement (CPU et RAM) avec une marge importante. Les répertoires de données proviennent d'un SAN et sont extensibles, principalement pour w.c.s. et fargo (porte-documents).

Caractéristiques d'une machine virtuelle à l'initialisation :

- processeur x86-64 4 cœurs
- mémoire vive 4Go
- espace disque 16Go ; et plus (via SAN) sur les applications stockant des documents w.c.s. et fargo.

Le serveur PostgreSQL est sécurisé, c'est-à-dire qu'il dispose au moins d'un slave pour *fail-over*. Si besoin il peut être installé par Entr'ouvert.

Accès à un S.I. tiers

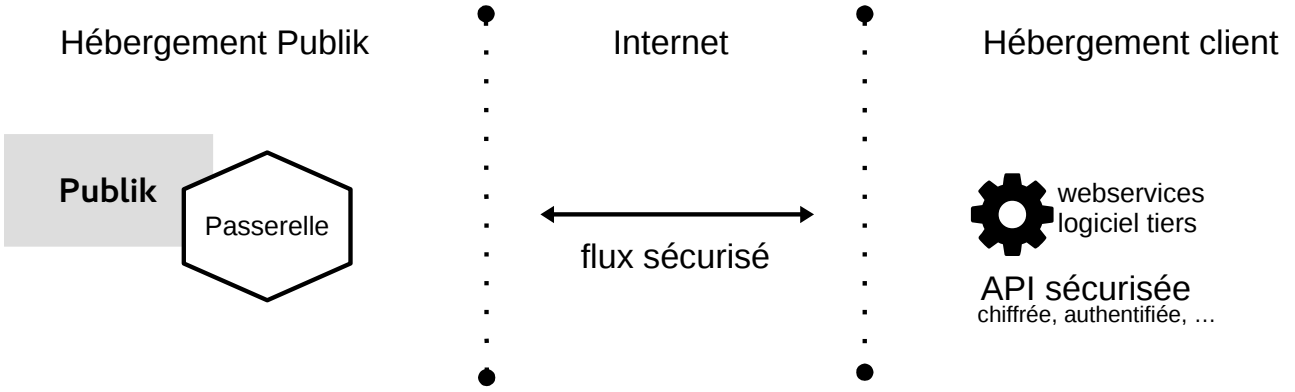
Lorsque Publik doit accéder à un système d'information tiers, il doit y avoir ouverture des flux nécessaires :

- accès webservices : passerelle consomme ou fourni des webservices au S.I. tiers ;
- accès LDAP pour les annuaires : accès par l'IdP pour authentification et synchonisation.

Pour les webservices, la connexion sera effectuée par Passerelle, via un connecteur.

Si le connecteur n'existe pas encore et doit donc être programmé, les webservices tiers doivent utiliser des protocoles ouverts et reconnus, tels que REST/JSON ou SOAP. Ils doivent être documentés et validés.

1 - Protection des accès webservices



Les webservices, en entrée comme en sortie, doivent utiliser HTTPS. L'authentification peut être :

- en login/mot de passe (*HTTP Basic authentication*)
- par certificat X509 client et serveur
- ... tout autre mode d'authentification peut être étudié par Entr'ouvert.

Note : les API internes à Publik utilisent un système d'authentification spécifique similaire à JWT, décrit dans http://doc.entrouvert.org/wcs/dev/#api

Note 2 : En cas de proxy sur la chaîne, il faut vérifier l'absence de cache.

2 - Protection de l'accès LDAP : X509

La connexion LDAP doit se faire en TLS, avec des certificats X509 clients et serveurs validés de chaque côté. Entr'ouvert peut fournir des certificats depuis son AC privée.

S'il s'agit d'un accès LDAP à un système Active Directory, voici deux documentation concernant la mise en place de TLS sur ce système :

- http://social.technet.microsoft.com/wiki/contents/articles/2980.ldap-over-ssl-ldaps-

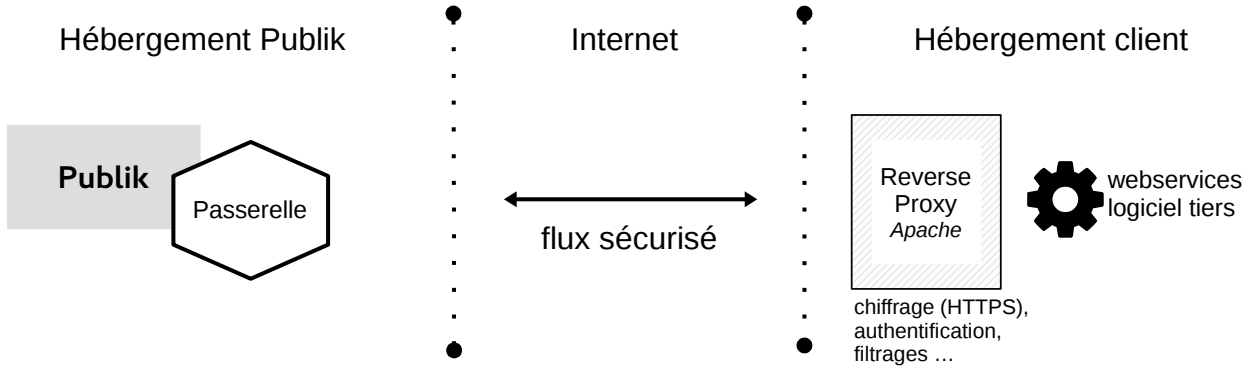
certificate.aspx
- http://www.javaxt.com/Tutorials/Windows/How_to_Enable_LDAPS_in_Active_Directory

3 - Ajout de protections sur webservices existants

Au cas où les logiciels tiers ne mettent pas en place de protection suffisante sur leurs webservices, plusieurs solutions de sécurisation peuvent être envisagées, dont les plus classiques sont : ajout d'un reverse-proxy, ajout d'une passerelle « locale », mise en place d'un VPN.

3.1 - Ajout d'un *reverse-proxy*

Un reverse-proxy est placé en frontal devant les webservices, ajoutant la couche HTTPS et/ou une authentification (par HTTP Basic ou X509). C'est généralement la solution la plus efficace, simple à mettre en place et ne nécessitant qu'une maintenance classique, qui peut être intégrée à la maintenance générale du SI. C'est donc la solution conseillée.



Le reverse-proxy est une solution de type Apache ou nginx. Il est connecté d'une part au webservice à diffuser, et d'autre part à un réseau accessible par Publik. Il ajoute :

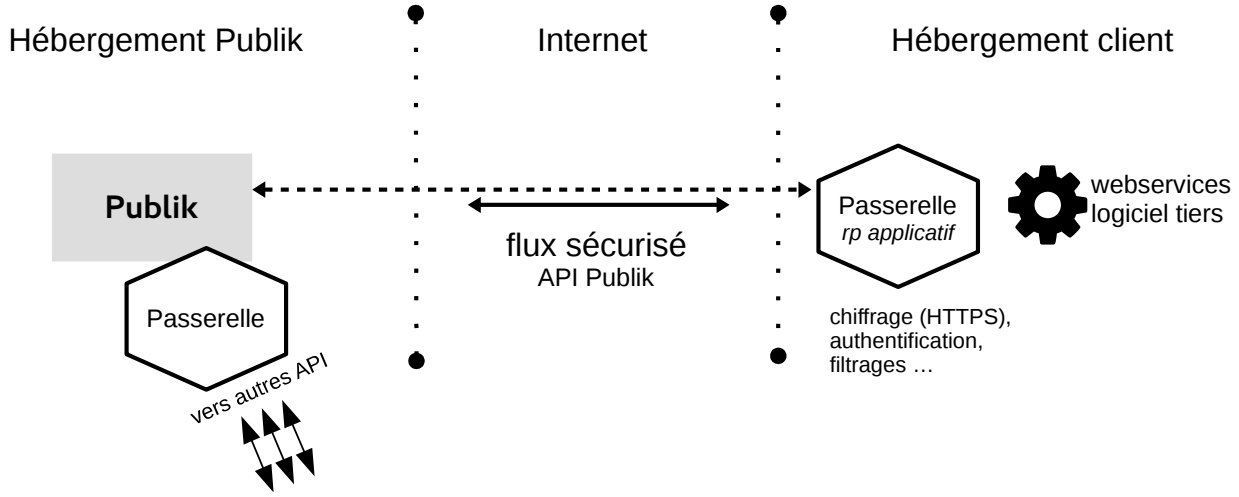
- le chiffrage du flux (HTTPS) ;
- une authentification : soit basique, soit basée sur un certificat X509 client ;
- un filtrage des URLs accessibles ou non depuis telle ou telle IP (typiquement pour l'IP de la passerelle Publik).

Un filtrage IP général peut également être ajouté au niveau d'un firewall placé en amont du reverse-proxy, par exemple le firewall d'arrivée générale du site client. Ce filtrage n'autorisera que l'IP de la passerelle Publik à accéder aux webservices.

3.2 - Ajout d'une instance passerelle « locale »

Dans cette configuration, une instance de passerelle est ajoutée, au même niveau qu'un reverse-proxy (*cf* supra). Les connecteurs de passerelle assurent alors localement la traduction des webservices tiers en webservices Publik, ces derniers ajoutant la sécurisation de l'accès entre le SI et Publik.

Cette solution présente l'avantage d'assurer un contrôle de sécurité fort des webservices diffusés à destination de Publik, contrôle assuré par Publik. Cependant, la maintenance est plus complexe que l'installation d'un reverse-proxy. Aussi ne doit-elle être mise en place que si les webservices de l'application sont très difficiles à sécuriser.



3.3 - Accès via VPN

Si la seule solution est un VPN, cela demande étude préalable par les équipes techniques d'Entr'ouvert (OpenVPN est le seul VPN déjà utilisé par Entr'ouvert en production).