entr'ouvert

publik

le connecteur citoyen

Architecture technique

Documentation

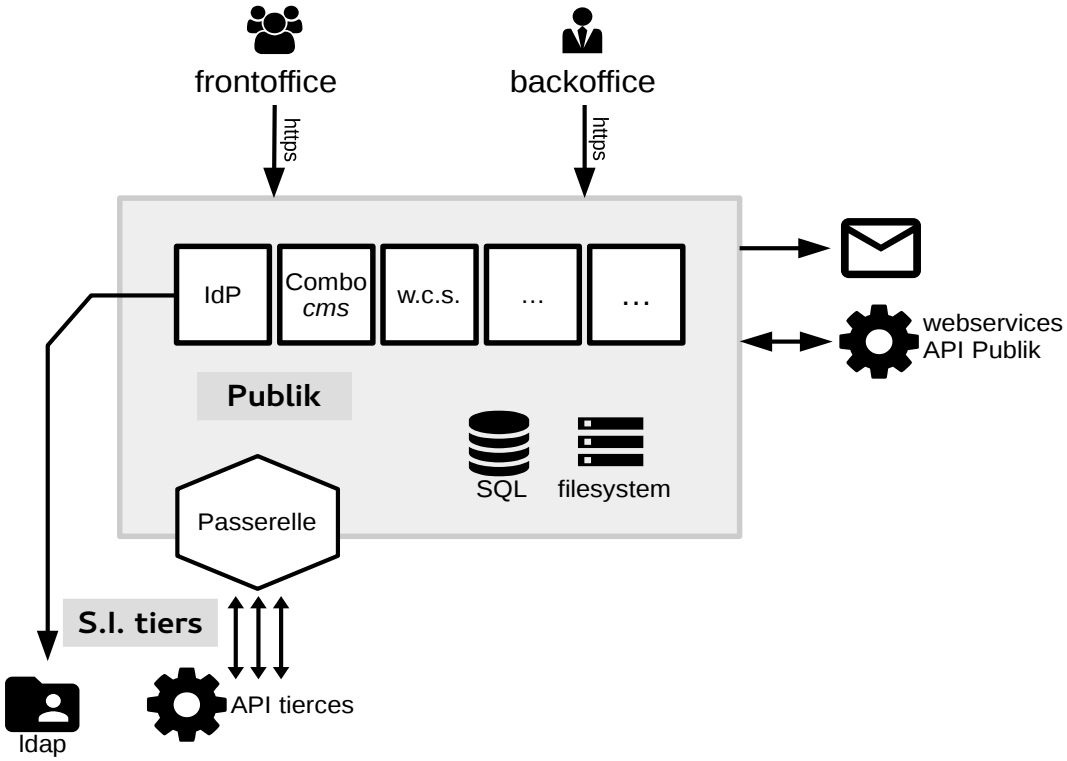
Table des matières

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 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 représentés 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 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 d'une ou plusieurs bases de données distincte.

4 - Flux réseau

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.

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 - Support et maintenance

Le support et la maintenance du système par Entr'ouvert nécessite un accès SSH avec les droits root (via su ou sudo). Entr'ouvert peut fournir une liste d'adresses IPv4 à ouvrir pour SSH.

Entr'ouvert n'assure le support que sur des machines Debian GNU/Linux.

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 :

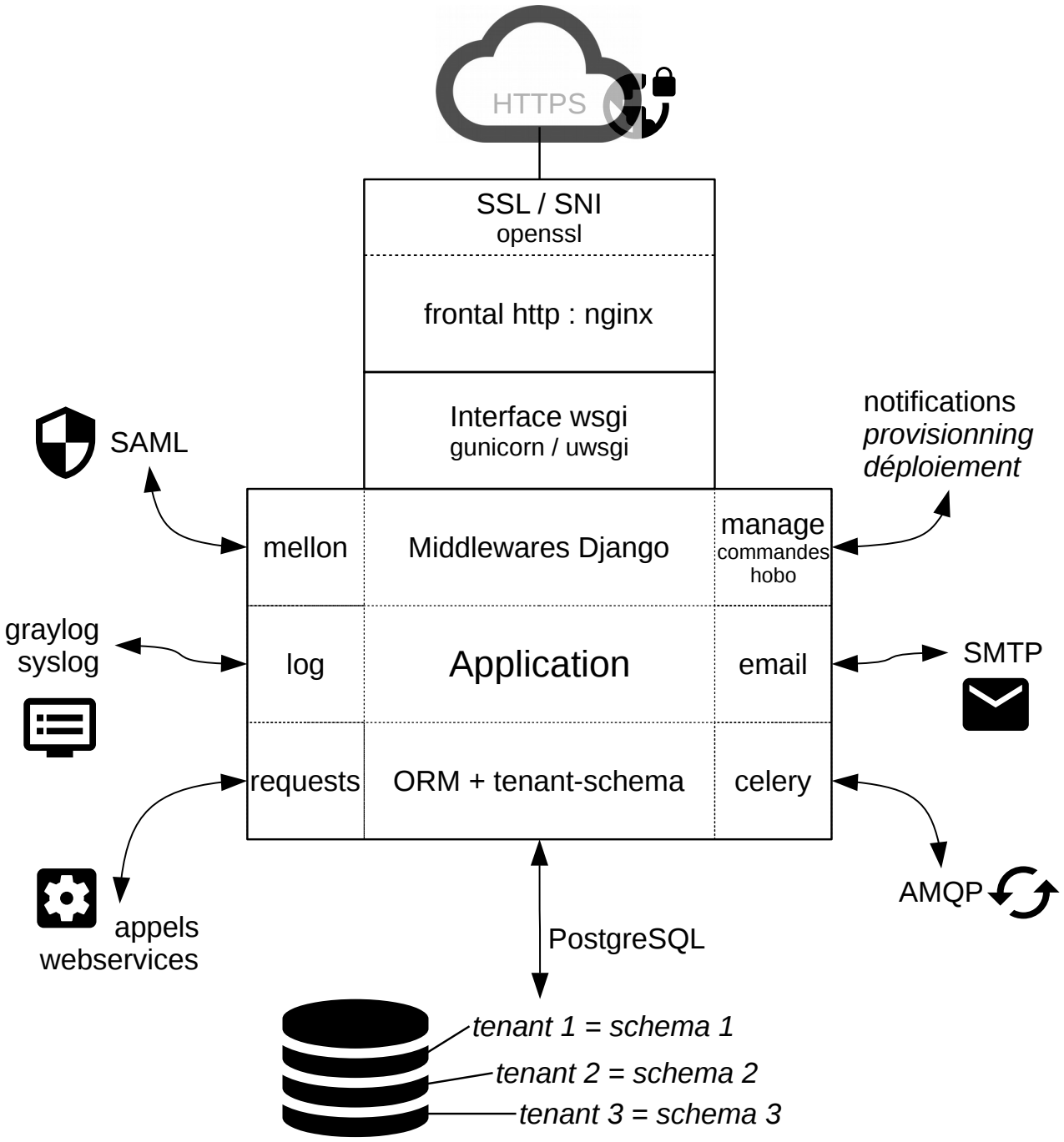
- recoit les données, les interprète et les valide avant de les envoyer à l'application ;
- permet de contrôler l'envoi des données aux bases de données par l'application (pas de requêtes SQL directes) ;
- sécurise les sorties de l'application (HTML) en imposant un contrôle fort sur les données affichable.

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 isolées 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 donnée sous forme d'un schema 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



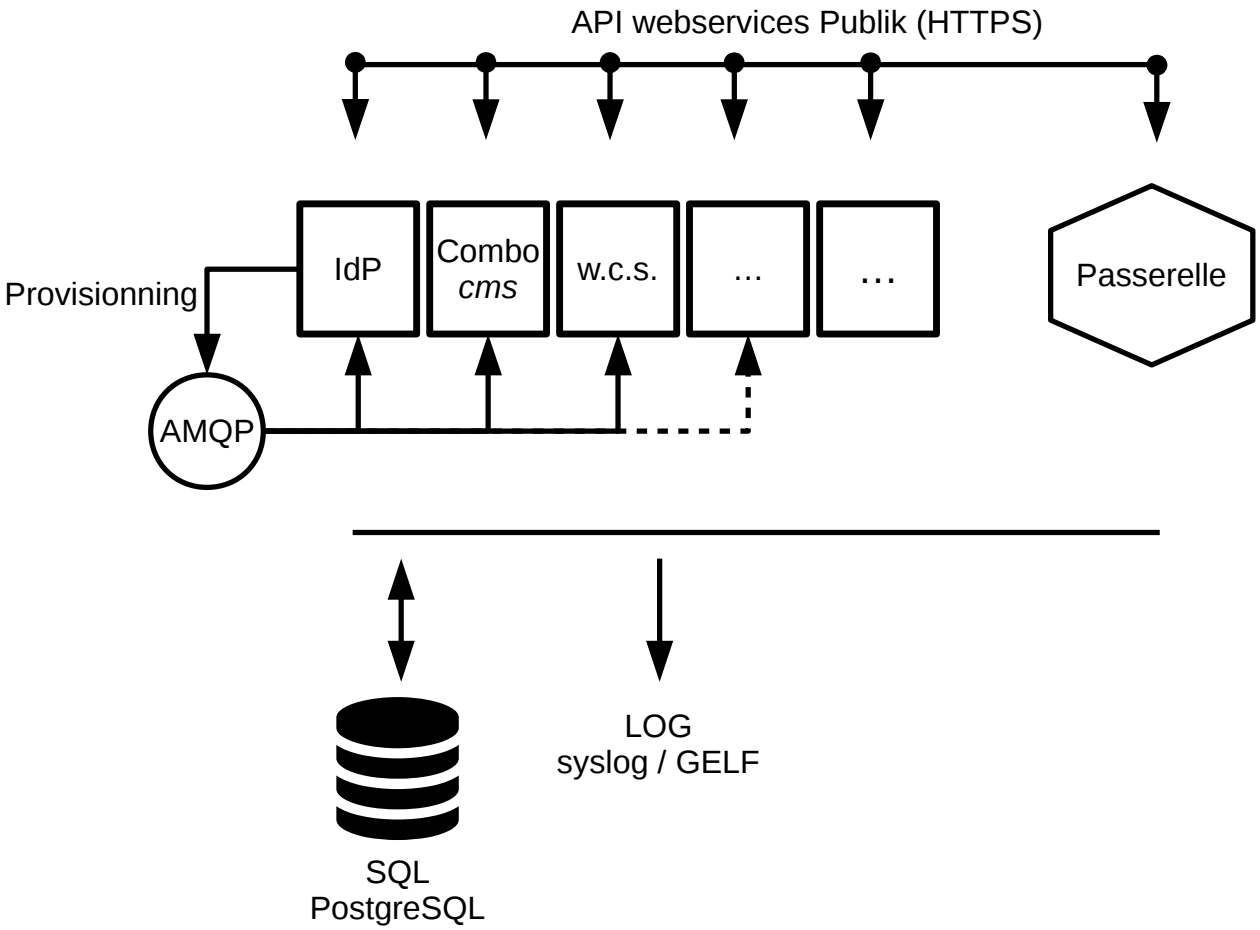
Le schéma ci-dessus montre que l'application n'est pas en contact « direct » avec l'extérieur. Elle utilise toujours des composants logiciels soit éprouvés (celery, requests, tenant-schemas) soit communs à toutes les briques (hobo, mellon).

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

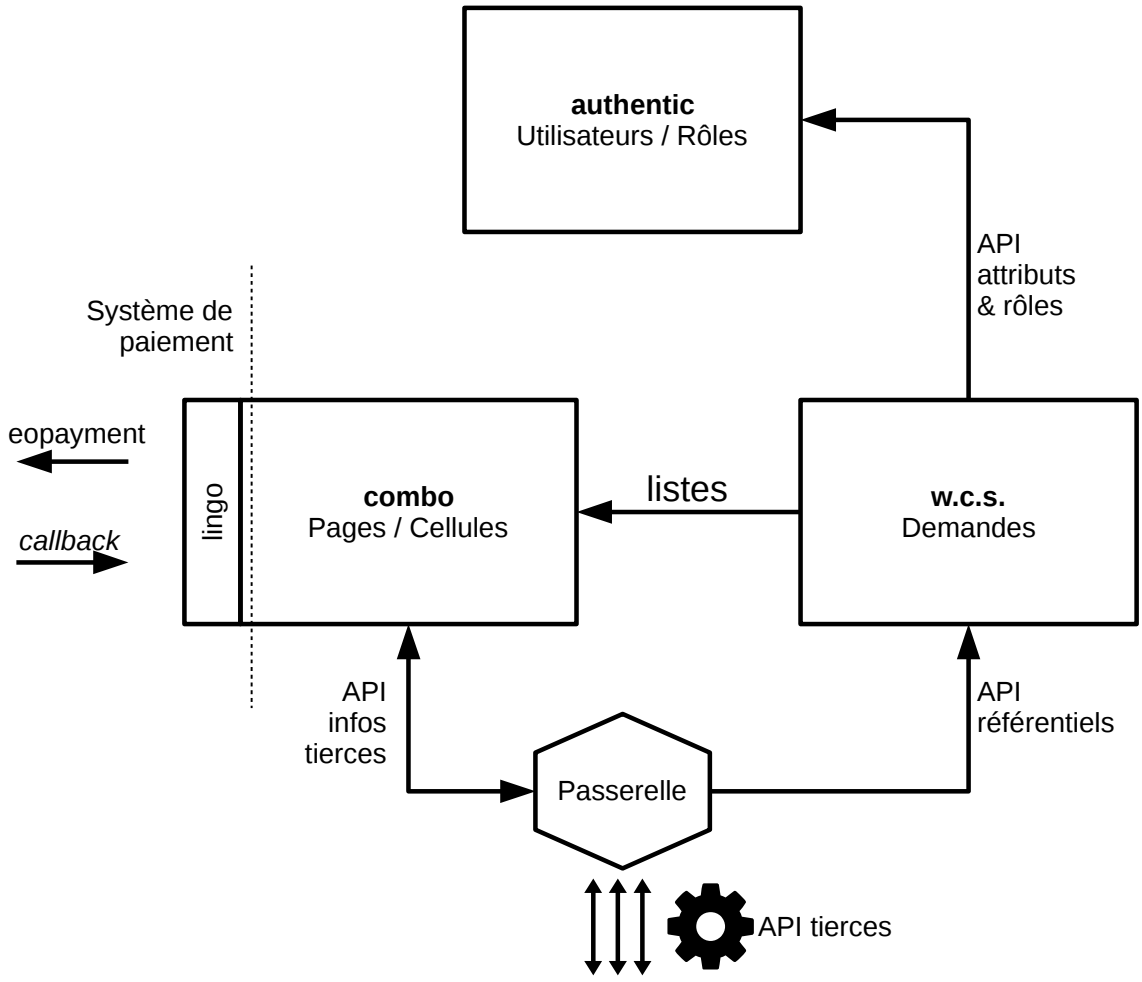
L'API des webservices Publik permet aux briques d'échanger des données. La plupart ont trait à l'échange de données autour d'un usager.

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).

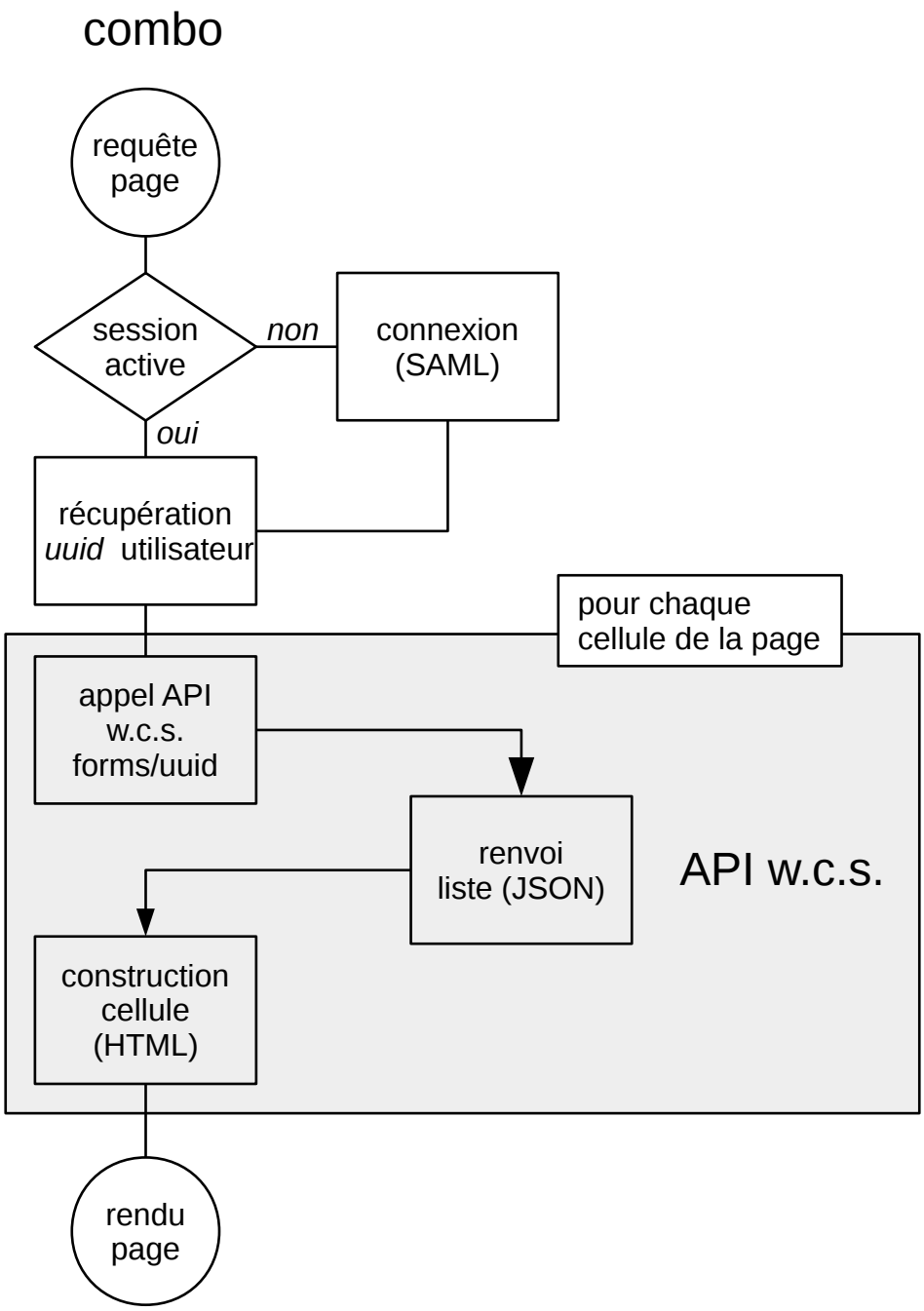
Consommateurs d'API :

- 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).



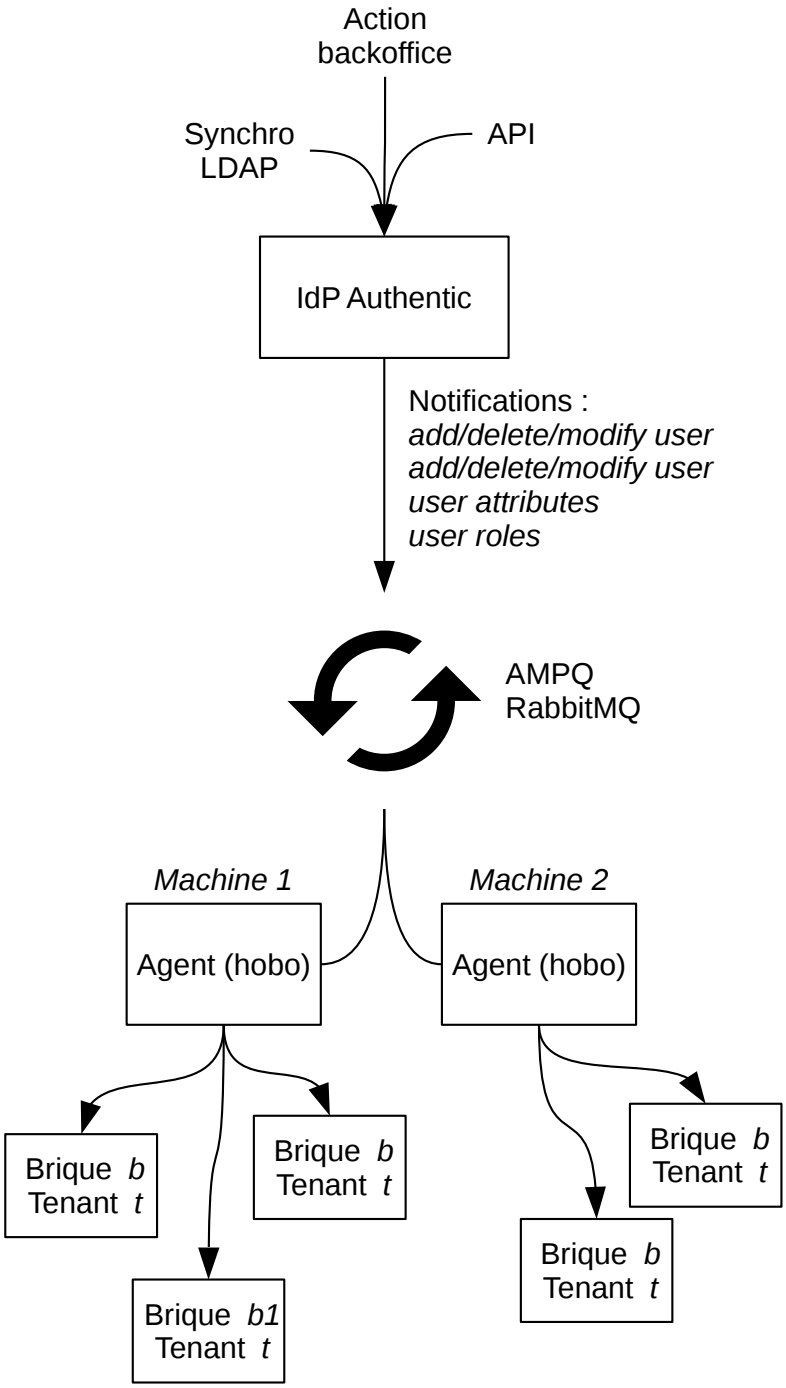
1.2 - Exemple d'un dialogue combo / w.c.s.

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 :

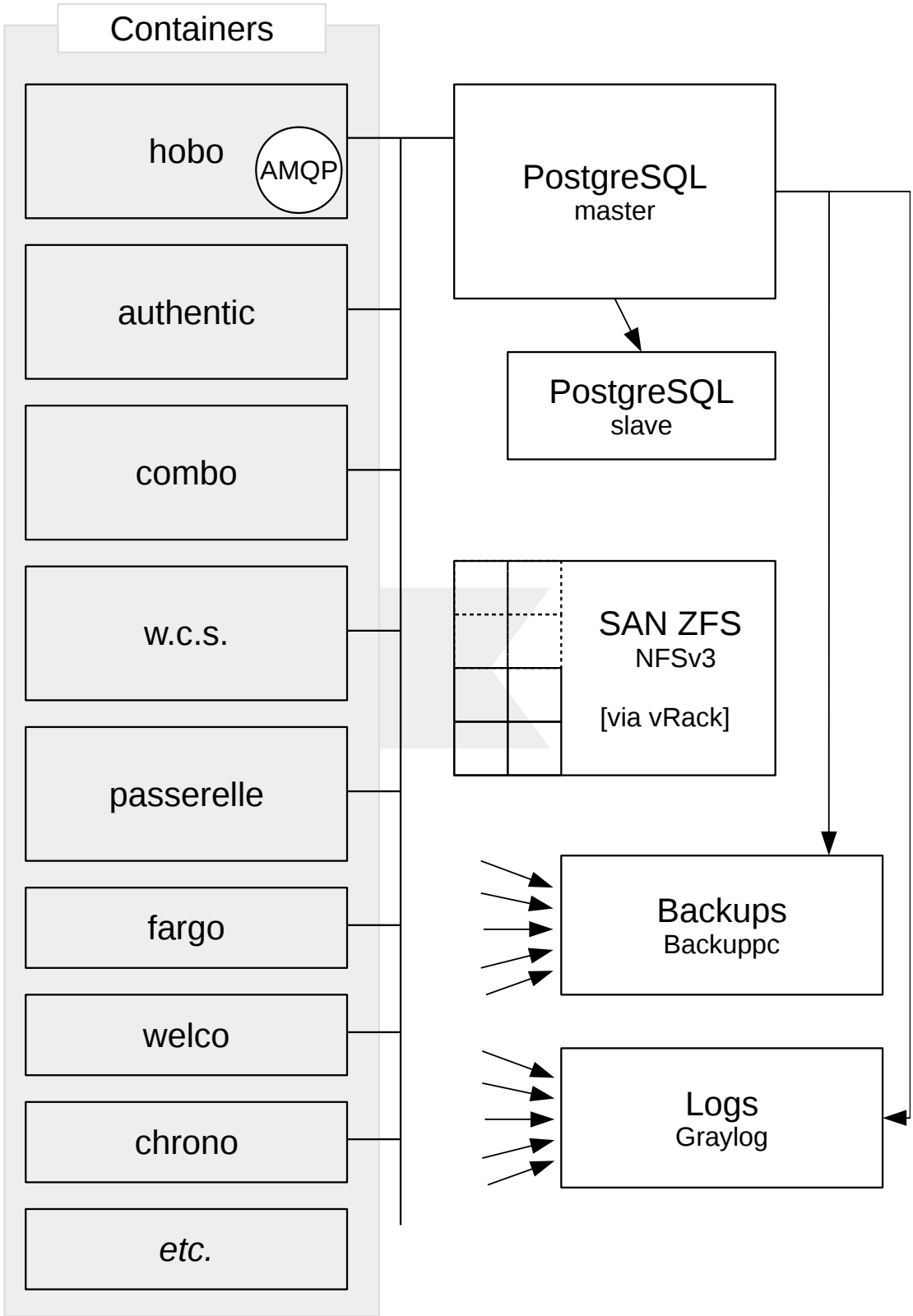
- En plus des snapshots ZFS, des backups incrémentaux (quotidiens) et complets (hebdomadaires) sont effectués sur une machine isolée hors site ;
- Le système de restauration est régulièrement vérifié ;
- La duplication d'une machine ou son transport vers un autre hôte physique peut être effectuée en 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'installation, 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 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.