Note: L'utilisation de cette page est dépréciée, consulter "la nouvelle documentation de référence": https://doc-publik.entrouvert.com/guide-de-l-administrateur-systeme/installation/pre-requis/

Installation d'une machine Publik sur base Debian 8 (Jessie)

Mode d'emploi pour l'installation d'une machine Publik. Il s'agit d'installer la solution sur une seule machine (physique ou virtuelle).

Cependant, sur le canevas de cette documentation, chaque composant peut également être installé sur une machine dédiée : il faut alors ajuster les règles de pare-feu et les connexions AMQP (exercice laissé au lecteur à ce stade).

Prérequis

DNS

Dans la zone DNS choisie, ajouter les noms des différentes briques:

```
publik          A     a.b.c.d    ; addresse IP de «publik»
portail         CNAME publik     ; portail usage (brique: combo)
backoffice      CNAME publik     ; portail agent (brique: combo)
connexion       CNAME publik     ; fournisseur d'identités (brique: authentic)
demarches       CNAME publik     ; téléservices (brique: wcs)
passerelle      CNAME publik     ; hub de webservices (brique: passerelle)
hobo            CNAME publik     ; système de déploiement (brique: hobo)
```

Dans la suite du document, l'installation est faite dans la zone example.net

Certificat x509

Publik ne fonctionne **qu'en mode HTTPS**.

Un ou plusieurs **certificats x509 valides et reconnus** doivent être disponibles qui couvrent tous les noms des briques qui seront installées.

Dans la suite de la documentation, un certificat est disponible :

- clé publique certifiée : /etc/ssl/certs/cert-example.pem
- clé privée : /etc/ssl/private/cert-example.key

Messagerie SMTP

Idéalement la machine doit disposer d'un SMTP local capable d'émettre des courriels. L'installation et configuration de exim4-daemon-light est en général suffisante.

Installation et paramétrage du système d'exploitation Debian 8 (Jessie)

Le système d'exploitation Debian 8.x (Jessie) est installé en mode minimal, i.e. uniquement le choix "utilitaires usuels du système" lors de la sélection des logiciels durant l'installation.

Ajout des depots

Dans les sources APT, ajouter les backports et le depôt Entr'ouvert :

```
# /etc/apt/sources.list

# Debian 8
deb http://httpredir.debian.org/debian/ jessie main
deb http://security.debian.org/ jessie/updates main
deb http://ftp.fr.debian.org/debian jessie-updates main

# Debian 8 backports
```

```
deb http://ftp.fr.debian.org/debian jessie-backports main

# Entr'ouvert
# key: wget -q -O- https://deb.entrouvert.org/entrouvert.gpg | apt-key add -
deb http://deb.entrouvert.org/ jessie main
```

Installation des composants de base

Publik utilise les composants suivants :

Framework Web Django

Installer la version 1.8 de jessie-backports, en s'assurant au préalable que le depot est bien présent dans sources.list

```
# apt install -t jessie-backports python-django
```

Serveur web nginx

```
# apt install nginx-full
```

Autoriser l'upload des "gros" fichiers: dans /etc/nginx/conf.d créer un fichier client-max-body-size.conf avec le contenu:

```
client_max_body_size 10M;
```

Taille à ajuster en fonction de l'usage prévu.

Les vhosts pour chaque composant Publik seront configurés en utilisant le modèle :

```
server {
    listen 443 ssl;
    server_name <hostname>;    # indiquer ici le ou les noms des instances prévues

    ssl_certificate /etc/ssl/certs/cert-example.pem;       # un certificat couvrant tous les noms
des instances prévues
    ssl_certificate_key /etc/ssl/private/cert-example.key;

    access_log /var/log/nginx/<hostname>-access.log combined;
    error_log /var/log/nginx/<hostname>-error.log;

    location ~ ^/static/(.+)$ {
        root /;
        try_files /var/lib/<application>/tenants/$host/static/$1
                  /var/lib/<application>/tenants/$host/theme/static/$1
                  /var/lib/<application>/collectstatic/$1
                  =404;
        add_header Access-Control-Allow-Origin *;
    }

    location ~ ^/media/(.+)$ {
        alias /var/lib/<application>/tenants/$host/media/$1;
    }

    location / {
        proxy_pass         http://unix:/run/<application>/<application>.sock;
        proxy_set_header   Host $http_host;
        proxy_set_header   X-Forwarded-SSL on;
        proxy_set_header   X-Forwarded-Protocol ssl;
        proxy_set_header   X-Forwarded-Proto https;
        proxy_set_header   X-Real-IP       $remote_addr;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}

# catchall http → https
```

```
server {
    listen   80;
    server_name  <hostname>;    # indiquer ici le ou les noms des instances prévues
    access_log  /var/log/nginx/<hostname>-access.log combined;
    error_log  /var/log/nginx/<hostname>-error.log;
    return 301 https://$host$request_uri;
}
```

ou <hostname> est le nom de domaine et <application> le nom de l'application deployée.

Base de données PostgreSQL

```
# apt install postgresql
```

En tant qu'utilisateur postgres créer les comptes et les bases pour chaque composant :

```
postgres@publik8$ createuser -SDR hobo ; createdb -O hobo hobo
postgres@publik8$ createuser -SDR combo ; createdb -O combo combo
postgres@publik8$ createuser -SDR passerelle ; createdb -O passerelle passerelle
postgres@publik8$ createuser -SDR fargo ; createdb -O fargo fargo
postgres@publik8$ createuser -SDR chrono ; createdb -O chrono chrono
postgres@publik8$ createuser -SDR bijoe ; createdb -O bijoe bijoe
postgres@publik8$ createuser -SdR wcs ; createdb -O wcs wcs_demarches_example_net
postgres@publik8$ createuser -SDR authentic-multitenant ; createdb -O authentic-multitenant authen
tic2_multitenant
```

Messagerie RabbitMQ

```
# apt install rabbitmq-server
```

Configuration de RabbitMQ

Prendre modèle sur l'exemple fourni avec hobo :

```
# zcat /usr/share/doc/rabbitmq-server/rabbitmq.config.example.gz > /etc/rabbitmq/rabbitmq.config
```

Modifications des indications SSL dans /etc/rabbitmq/rabbitmq.config :

```
…
   {ssl_options, [{certfile,            "/etc/ssl/certs/cert-example.pem"},
                  {keyfile,             "/etc/ssl/private/cert-example.key"},
                  {versions, ['tlsv1.2', 'tlsv1.1']}]}
…
   {listener, [{port,     15672},
               {ssl,      true},
               {ssl_opts, [{certfile,   "/etc/ssl/certs/cert-example.pem"},
                           {keyfile,    "/etc/ssl/private/cert-example.key"},
                           {versions, ['tlsv1.2', 'tlsv1.1']}]}]}
…
```

Puis relancer le service :

```
root@publik8# service rabbitmq-server restart
```

Puis ajouter un utilisateur hobo dans rabbitmq (choisir un mot de passe complexe) :

```
root@publik8# rabbitmqctl add_user hobo <mot-de-passe-complexe>
root@publik8# rabbitmqctl set_permissions hobo ".*" ".*" ".*"
```

Les composants Publik

Système de déploiement hobo

Installation du serveur :

```
# apt install hobo
```

Lancement de hobo (provoque l'initialisation de la base) :

```
root@publik8# service hobo restart
```

Agent de déploiement hobo-agent

Les instances cibles seront déployés sur la machine elle-même, on installe donc l'agent localement :

```
# apt install hobo-agent
```

Configurer l'agent afin qu'il puisse se connecter au serveur rabbitmq. Dans /etc/hobo-agent/settings.py, modifier la variable BROKER_URL :

```
BROKER_URL = 'amqp://hobo:<mot-de-passe-complexe>@<hobo hostname>:5671//?ssl=1'
```

Pus lancer le service hobo-agent via supervisor (à la première installation, ce dernier est démarré avant l'installation de hobo-agent) :

```
# supervisorctl update
# supervisorctl restart hobo-agent
```

Fournisseur d'identités Authentic

Installation :

```
# apt install authentic2-multitenant
```

Ajouter une configuration nginx, en se basant sur le modèle et remplaçant <application> par authentic2-multitenant :

Activation du provisionning

Authentic va utiliser RabbitMQ pour diffuser les informations sur les utilisateurs et les rôles. Pour cela, ajouter à la fin du fichier /etc/authentic2-multitenant/config.py :

```
# extrait de /etc/authentic2-multitenant/config.py
# (une ligne ajoutée fin du fichier)

# Role provisionning via local RabbitMQ
HOBO_ROLE_EXPORT = True
```

Puis relancer le service pour prendre en compte la nouvelle configuration :

```
# service authentic2-multitenant restart
```

Authentification via FranceConnect

Afin de pouvoir utiliser l'authentification via FranceConnect il faut installer le paquet supplémentaire python-authentic2-auth-fc:

```
# apt install python-authentic2-auth-fc
```

Portails citoyen et agent Combo

Installation :

```
# apt install combo
```

Création d'une configuration nginx en se basant sur le modèle et remplaçant <application> par combo.

Combo est utilisé pour les portails usager et agent, le <hostname> couvrira les deux sites, par exemple :

```
# extrait de /etc/nginx/sites-available/combo
server {
…
    server_name portail-agent.example.net portail-usager.example.net;
…
}
```

Démarches w.c.s.

Installation du paquet wcs-au-quotidien :

```
# apt install wcs-au-quotidien
```

Un ensemble de paramètres, spécifiés dans des fichiers de configuration peut-être défini lors de la création d'une instance w.c.s. Un archive zip contenant ses fichiers de configuration doit être placée dans le répertoire /var/lib/wcs/skeletons. Par exemple /var/lib/wcs/skeletons/publik.zip (fichier joint à cette page).

Configuration nginx :

```
# /etc/nginx/sites-available/wcs
server {
    listen 443 ssl;
    server_name <hostname>;

    ssl_certificate /etc/ssl/certs/cert-example.pem;
    ssl_certificate_key /etc/ssl/private/cert-example.key;

    access_log /var/log/nginx/wcs-access.log combined;
    error_log /var/log/nginx/wcs-error.log;

    location ~ ^/static/(.+)$ {
            root /;
            try_files /var/lib/wcs/$host/static/$1
                      /var/lib/wcs/$host/theme/static/$1
                      /var/lib/wcs/collectstatic/$1
                      /var/lib/wcs-au-quotidien/collectstatic/$1
                      =404;
    }

    location /qo { alias  /usr/share/wcs/qommon/; }
    location /apache-errors { alias  /usr/share/auquotidien/apache-errors/; }
    location /themes {
        root /;
        try_files /var/lib/wcs/$host$uri
                  /var/lib/wcs-au-quotidien/$host$uri
                  /usr/share/wcs/$uri
                  =404;
    }

    location /robots.txt {
        alias /usr/local/share/auquo-robots.txt;
    }

    location /rpc_relay.html {
```

```
        alias /usr/share/auquotidien/rpc_relay.html;
    }

    location / {
        proxy_pass          http://unix:/run/wcs/wcs.sock;
        proxy_set_header    Host $http_host;
        proxy_set_header    X-Forwarded-SSL on;
        proxy_set_header    X-Forwarded-Protocol ssl;
        proxy_set_header    X-Forwarded-Proto https;
        proxy_set_header    X-Real-IP       $remote_addr;
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    }

}

# catchall http → https
server {
    listen   80;
    server_name  <hostname>;
    access_log  /var/log/nginx/wcs-access.log combined;
    error_log  /var/log/nginx/wcs-error.log;
    return 301 https://$host$request_uri;
}
```

Hub de webservices Passerelle

Installation :

```
# apt install passerelle
```

Création d'une configuration nginx à partir du modèle en remplaçant <application> par passerelle.

Porte document Fargo

Installation :

```
# apt install fargo
```

Création d'une configuration nginx à partir du modèle en remplaçant <application> par fargo.

Thèmes de base

Un ensemble de thèmes est fourni par le paquet publik-base-theme, donc le thème de base «publik» utile pour déployer des instances initialement (sans thème spécifique).

Installation :

```
# apt install publik-base-theme
```

Création des instances : hobo «cooking»

Créer un fichier «recipe.json» (dans /tmp par exemple) sur ce modèle :

```
{
  "variables": {
    "hobo": "hobo.example.net",
    "authentic": "connexion.example.net",
    "combo": "portail.example.net",
    "combo_agent": "backoffice.example.net",
    "passerelle": "passerelle.example.net",
    "wcs": "demarches.example.net",
    "fargo": "portdoc.example.net"
```

```
  },
  "steps": [
    {"create-hobo": {
      "url": "https://${hobo}/"
    }},
    {"create-superuser": {
      "email": "superuser@example.net",
      "password": "example"
    }},
    {"create-authentic": {
      "url": "https://${authentic}/",
      "title": "Connexion"
    }},
    {"set-idp": {
    }},
    {"create-combo": {
      "url": "https://${combo}/",
      "title": "Compte citoyen",
      "template_name": "portal-user"
    }},
    {"create-combo": {
      "url": "https://${combo_agent}/",
      "slug": "portal-agent",
      "title": "Portail agent",
      "template_name": "portal-agent"
    }},
    {"create-wcs": {
      "url": "https://${wcs}/",
      "title": "Démarches",
      "template_name": "publik.zip"
    }},
    {"create-passerelle": {
      "url": "https://${passerelle}/",
      "title": "Passerelle"
    }},
    {"create-fargo": {
      "url": "https://${fargo}/",
      "title": "Porte doc"
    }},
    {"set-theme": {
      "theme": "publik"
    }}
  ]
}
```

L'utiliser sur la commande cook de hobo-manage (à lancer en tant qu'utilisateur hobo !) :

```
# sudo -u hobo hobo-manage cook /tmp/recipe.json
```

Attention : la commande cook arrête d'envoyer des instructions de déploiement au bout d'un *timeout* de 30 secondes. Sur des systèmes peu véloces, il faudra éventuellement relancer la commande plusieurs fois, jusqu'à ce que son exécution se déroule sans aucune erreur.

Finalisations

Passage de w.c.s. en SQL

!! Cette étape n'est pas nécessaire si le deploiement de l'instance w.c.s a été fait en utilisant un template de configuration (variable template_name dans le fichier recipe) activant le stockage des données en SQL.

Le système de déploiement n'utilise pas automatiquement le stockage SQL de w.c.s. Il faut donc, une fois l'instance créée, la convertir manuellement. Pour cela :

1. Ajouter « postgresql = true » dans la section « options » du fichiers /var/lib/wcs/demarches.example.net/site-options.cfg :

```
# extrait de /var/lib/wcs/demarches.example.net/site-options.cfg

[options]
postgresql = true
# …  autres options laissées inchangées
```

2. Lancer la conversion proprement dite (en tant qu'utilisateur wcs-au-quotidien) :

```
# sudo -u wcs wcsctl -f /etc/wcs/wcs-au-quotidien.cfg convert-to-sql --dbname=wcs_demarches_ex
ample_net demarches.example.net
```

Amorce de la configuration de w.c.s. par création d'un premier utilisateur

Se rendre sur https://connexion.example.net/manage/roles/ et créer un premier rôle nommé par exemple «Support et Débogage»

Se rendre sur https://connexion.example.net/manage/users/ et créer un utilisateur :

- dans la collectivité «Collectivité par défaut»
- avec le statut super-utilisateur

Après quelques secondes, vérifier que l'utilisateur est bien provisionné sur w.c.s. https://demarches.example.net/backoffice/users/ et qu'il a bien les rôles «Administrateur du site» et «Support et Débogage»

Fin ... et début !

L'installation est prête. Il reste à effectuer le paramétrage de base du système puis à commencer la configuration de Publik.

Fichiers

publik.zip	694 octets	31 octobre 2017	Serghei Mihai