Hobo - Development #43245

revoir les mécaniques de provisionning

24 mai 2020 16:31 - Frédéric Péters

Statut:	Fermé	**Début:**	24 mai 2020
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Aujourd'hui Autehntic(/hobo) envoie des messages en broadcast sur rabbitmq, ceux-ci sont reçus/interprétés par des agents hobo (1 par serveur) qui vont lancer des commandes xxx-manage hobo_notify < les-infos.

C'est parfois compliqué à débugguer, en partie parce qu'on manque d'outils d'analyse. (ex: #40214)

Pour éviter que les hobo_notify ne s'exécutent en parallèle (#8374), on tourne avec un seul worker. Ça fait un goulot d'étranglement qui se sent de plus en plus.

Le même bus/worker est utilisé pour les messages des provisionnings des services, qui sont des messages dont le traitement peut être plus long. On se trouve là du coup à relancer des messages dans le vide, au cas où les premiers ne seraient pas passés (#42956).

~~

De là, on peut avoir des plans, par exemple se dire qu'on pourrait implémenter SCIM qui ne servirait pour du provisionning avec des services extérieurs, et qu'on pourrait alors l'utiliser également en interne. Ou inverser la logique actuelle et avoir les services qui sur un signal demandent des infos à l'IdP.

Mais pour changer le moins de choses, j'aurais comme proposition que les différentes applications exposent un endpoint de provisionning (mettons /__provision__/) qui acceptent un POST avec les payload qu'on a actuellement dans les messages hobo-notify envoyés sur rabbitmq.

En pratique, il me semble :

- ça permet de réutiliser 90% du code d'hobo/agent/common/management/commands/hobo_notify.py
- pour l'envoi des messages ça demande de modifier le notify_agents() pour faire de l'HTTP plutôt que poser le message dans le rabbitmq local, ça n'empêche pas ici de réfléchir à des moyens asynchrones d'assurer ça par authentic, rabbitmq local quand même, mule uwsgi, whatever.

Avantages immédiats :

- des messages HTTP, on est armé pour les gérer.
- chaque tenant est indépendant, le provisionning n'est pas ralenti partout parce qu'une synchro LDAP a lieu quelque part.
- parallèle au déploiement des services, pas de ralentissement parce qu'un site se déploie.
- insensible à la manière dont le service a été déployé (e.g. ça corrige #43192 tout seul).
- très proche du code qu'on a déjà aujourd'hui.

Demandes liées:

Lié à Authentic 2 - Development #70751: synchro/provisionning : être capable ...	**Nouveau**	**27 octobre 2022**

Révisions associées

Révision 00127616 - 29 mai 2020 12:24 - Frédéric Péters

general: use HTTP API to provision users & groups (#43245)

Historique

#1 - 25 mai 2020 21:21 - Frédéric Péters

Pour illustrer, https://git.entrouvert.org/hobo.git/log/?h=wip/43245-provisionning-draft

#2 - 27 mai 2020 15:09 - Frédéric Péters

- Fichier 0001-general-use-HTTP-API-to-provision-users-groups-43245.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

Je laisse "draft" dans le nom de la branche mais c'est désormais ok.

C'est basiquement comme prévu, avec peut-être comme points à noter :

- dans le diff le gros truc qui apparait c'est le code de hobo_notify.py pour une grande part déplacé dans provisionning/utils.py, on peut me croire, ou inspecter les choses,
- c'est désactivé par défaut,
- ça s'active, par authentic, via HOBO_HTTP_PROVISIONNING = True
- j'avais parlé dans la description de mule uwsgi etc. mais le code authentic créait déjà un thread pour sortir du cycle requête/réponse, je laisse juste ça ainsi,
- une fois activé, si jamais le provisionning HTTP ne fonctionne pas pour certains services, il y a fallback sur l'AMQP,
- il y a un peu de nouveau code, pour gérer le cas particulier #43192 (provisionning vers hobo lui-même), c'est fait de manière totalement adhoc, il y aurait peut-être à un moment à réfléchir pour intégrer un fichier hobo.json et faire que le code "normal" (settings loaders etc.) soit exploité, mais ça sortait du cadre,
- il y a aussi le code côté wcs, https://jenkins.entrouvert.org/job/wcs-wip/job/wip%252F43245-provisionning-draft/

#3 - 29 mai 2020 10:23 - Benjamin Dauvergne

Je relis.

#4 - 29 mai 2020 11:10 - Benjamin Dauvergne

- Statut changé de Solution proposée à En cours

provisionning.py :

Toujours mon avis personnel, mais je serai plus à l'aise avec des opérations sur les set dans notify_agents et moins modifier data, ça enlève le else dans le try et les allers/retours sur data['audience'] :

```
            rest_audience = set(data['audience']) & set(services_by_url)
            amqp_audience = set(data['audience']) - set(services_by_url)
            data['audience'] = list(amqp_audience)
...
                        json=dict(data, audience=[audience]))
```

---

middleware.py:

Même avis que toi pour le code hobo ad-hoc, c'est moche mais ça sort du cadre du ticket, mais quand même pour ne pas qu'on oublie, serait-il possible de faire une méthode séparée de toute la partie if 'hobo.environement' par exemple self.hobo_specific_setup() avec un commentaire signalant notre volonté de voir sa mort prochaine ?

La fin du traitement devrait se faire dans un thread (tout le for i in range(20)), histoire de ne pas se retrouver avec des provisionning foirés à cause du harakiri ou du timeout nginx (et là avec des erreurs de notification coté authentic qui seront fausses si le provisionning s'est bien terminé avant le harakiri). Les seuls cas où authentic lèvera une erreur c'est sur une perte de connectivité, sur un problème de provisionning l'erreur ne viendra que de la brique et c'est mieux.

Pour parler :

Je regarde à passer directement request à get_local_dict() pour ne pas réfléchir au positionnement de StoreRequestMiddleware dans la liste (pas que ce soit compliqué, mais pour quelqu'un qui ne connaît pas le code ça rajoute de la complexité inutile au flot des données); en fait en tirant un peu la pelote, get_installed_services_dict() n'est utilisé que par get_hobo_json() et la vue debug_json, donc un seul cas où la requête est hors contexte (dans debug_json on peut passer request directement). Pour get_hobo_json :

- dans hobo.context_processor : la requête est disponible dans theme_base(), mais surtout hobo devrait avoir THEME_SKELETON_URL dans ses settings et aucune donnée utilisée ici ne dépend de la requête,
- dans hobo.deploy.signals: signal request_finished, de toute façon plus de requête dans StoreRequestMiddleware à ce moment (c'est après le passage des middleware)

Bon c'est pour plus tard mais je pense que ce serait une bonne chose de diminuer les usages de StoreRequestMiddleware dans le futur (certainement en faisant d'hobo un bon citoyen de Publik au niveau settings déjà).

#5 - 29 mai 2020 12:31 - Frédéric Péters

> Toujours mon avis personnel, mais je serai plus à l'aise avec des opérations sur les set dans notify_agents et moins modifier data, ça enlève le

> else dans le try et les allers/retours sur data['audience'] :

Je l'avais fait de manière similaire mais au final j'ai préféré le amqp_audience.remove() une fois qu'on sait que la requête est bien passée côté HTTP (pas uniquement sur le côté déclaratif de la présence de provisionning-url) (scénario : mise à jour et corbo pas mis à jour pas redémarré et il se trouve connu comme gérant provisionning-url par Hobo alors qu'en fait, pas).

> self.hobo_specific_setup()

Fait.

> La fin du traitement devrait se faire dans un thread

J'aime assez l'idée que non, qu'une erreur 500 si elle a lieu soit visible dans les logs nginx. La boucle est en fait là sur l'IntegrityError si jamais il y a deux provisionning du même utilisateur en même temps; j'imagine le cas bien rare, mais pour éviter l'harakiri, je pourrais changer cette boucle pour la limiter dans le temps, se donner disons très grand max 5 secondes pour l'appel. (en local les logs nginx m'indiquent 0,042 secondes pour le traitement de provisionning (combo)).

#6 - 29 mai 2020 14:13 - Benjamin Dauvergne

- Statut changé de En cours à Solution validée

Frédéric Péters a écrit :

> > Toujours mon avis personnel, mais je serai plus à l'aise avec des opérations sur les set dans notify_agents et moins modifier data, ça enlève le else dans le try et les allers/retours sur data['audience'] :
>
> Je l'avais fait de manière similaire mais au final j'ai préféré le amqp_audience.remove() une fois qu'on sait que la requête est bien passée côté HTTP (pas uniquement sur le côté déclaratif de la présence de provisionning-url) (scénario : mise à jour et corbo pas mis à jour pas redémarré et il se trouve connu comme gérant provisionning-url par Hobo alors qu'en fait, pas).

Ok.

> > La fin du traitement devrait se faire dans un thread
>
> J'aime assez l'idée que non, qu'une erreur 500 si elle a lieu soit visible dans les logs nginx. La boucle est en fait là sur l'IntegrityError si jamais il y a deux provisionning du même utilisateur en même temps; j'imagine le cas bien rare, mais pour éviter l'harakiri, je pourrais changer cette boucle pour la limiter dans le temps, se donner disons très grand max 5 secondes pour l'appel. (en local les logs nginx m'indiquent 0,042 secondes pour le traitement de provisionning (combo)).

Je ne pensais vraiment pas à des histoires concurrence compliquée (d'ailleurs le range(20) me parait exagéré avec le recul), plutôt l'ajout/retrait d'un rôle par héritage, l'injection d'un CSV ou une synchro LDAP fraîche où on peut avoir des centaines d'utilisateurs provisionnés d'un coup, mais si c'est vraiment 42ms par utilisateur effectivement ça passera dans les 120 secondes en général, jusqu'au jour où on injectera 5000 utilisateurs d'un coup; m'enfin le 504 au bout de 30s sera un peu inutile (on aura pas de 500, sauf bug dans le code de provisionning); 5 secondes c'est insuffisant pour l'ajout d'un rôle à tous les agents d'une grosse métro, et en espérant que uwsgi ne tue par le worker sur une socket fermée par nginx (je ne sais pas comment il se comporte dans ce cas).

#7 - 29 mai 2020 15:55 - Frédéric Péters

- Statut changé de Solution validée à Résolu (à déployer)

```
commit 0012761656db907e2c037c89ed694f4f76a76651
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Mon May 25 21:15:26 2020 +0200

    general: use HTTP API to provision users & groups (#43245)
```

#8 - 29 mai 2020 17:16 - Frédéric Péters

- Statut changé de Résolu (à déployer) à Solution déployée

#9 - 27 octobre 2022 08:48 - Paul Marillonnet

Frédéric Péters a écrit :

> De là, on peut avoir des plans, par exemple se dire qu'on pourrait implémenter SCIM qui ne servirait pour du provisionning avec des services extérieurs, et qu'on pourrait alors l'utiliser également en interne.

Hop, petit déterrage de ticket parce qu'une doc Azure détaillée existe enfin à ce sujet :
https://learn.microsoft.com/fr-fr/azure/active-directory/app-provisioning/use-scim-to-provision-users-and-groups

#12 - 27 octobre 2022 09:26 - Paul Marillonnet

- Lié à Development #70751: synchro/provisionning : être capable d'interroger/recevoir d'un fournisseur d'identité tiers des données selon le protocole SCIM ajouté

Fichiers

0001-general-use-HTTP-API-to-provision-users-groups-43245.patch	38,7 ko	27 mai 2020	Frédéric Péters