Gestion des accès

La gestion des accès dans authentic a deux objectifs:

- gérer les accès aux service soit directement soit indirectement en leur transmettant des attributs,
- gérer les accès à la partie administrative d'authentic lui même, en permettant une utilisation multi-tenant i.e. pouvoir gérer plusieurs organisation ayant éventuellement des relations hiérarchiques entre elles.

L'idée est de partir du modèle RBAC et d'en garder l'essentiel pour couvrir ces deux besoins.

Error executing the **plantuml** macro (Missing partial wiki_external_filter/_macro_image with {:locale=>[:en], :formats=>[:pdf], :variants=>[], :handlers=>[:raw, :erb, :html, :builder, :ruby, :rsb]}. Searched in: * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/redmine_tags/app/views" * "/usr/share/redmine/plugins/redmine_entrouvert/app/views" * "/usr/share/redmine/plugins/plantuml/app/views" * "/usr/share/redmine/plugins/localizable/app/views" * "/usr/share/redmine/app/views")

`Modèle RBAC avec héritage`

Nous n'avons pas besoin de la partie sur les permissions qui rend le modèle complexe à gérer; on confondra toujours un rôle avec une permission sur un objet, i.e. pour chaque paire permission, objet dont nous aurions besoin, il y aura un rôle équivalent qui pourra se combiner avec d'autres rôle en utilisant l'héritage permettant ainsi exactement les mêmes possibilités que le modèle RBAC complet.

Error executing the **plantuml** macro (Missing partial wiki_external_filter/_macro_image with {:locale=>[:en], :formats=>[:pdf], :variants=>[], :handlers=>[:raw, :erb, :html, :builder, :ruby, :rsb]}. Searched in: * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/redmine_tags/app/views" * "/usr/share/redmine/plugins/redmine_entrouvert/app/views" * "/usr/share/redmine/plugins/plantuml/app/views" * "/usr/share/redmine/plugins/localizable/app/views" * "/usr/share/redmine/app/views")

`Modèle RBAC simplifié pour Authentic`

Notion d'héritage

L'héritage entre rôle n'a rien à voir avec l'héritage en programmation. Si un rôle R1 hérite d'un rôle R2, cela veut dire que le rôle R1 possède en plus des siennes toutes les permissions associées au rôle R2, soit directement soit via héritage. C'est une relation transitive, si R2 hérite de R3 alors R1 hérite indirectement aussi de R3.

Cela revient aussi à dire que tous les membres de R1 sont aussi indirectement des membres de R2.

Les rôles

Les rôles seront séparés en deux:

- les rôles fonctionnels créés et gérés par les utilisateurs, ils ne sont lié directement à aucune permission sur aucun objet, ils peuvent seulement hériter d'autres rôles,
- les rôles techniques gérés par leur système, ils sont créés, gérés et supprimés par le système selon le besoin, pour chaque et chaque action pour laquelle il faudrait gérer une autorisation un tel rôle sera créé.

Error executing the **plantuml** macro (Missing partial wiki_external_filter/_macro_image with {:locale=>[:en], :formats=>[:pdf], :variants=>[], :handlers=>[:raw, :erb, :html, :builder, :ruby, :rsb]}. Searched in: * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/redmine_tags/app/views" * "/usr/share/redmine/plugins/redmine_entrouvert/app/views" * "/usr/share/redmine/plugins/plantuml/app/views" * "/usr/share/redmine/plugins/localizable/app/views" * "/usr/share/redmine/app/views")

`Modèle RBAC simplifié pour Authentic`

Rôles techniques d'administration

Pour permettre un découpage à gros grain de l'administration on ajoute le concept d'organisation, chaque objet (y compris les utilisateurs et les rôles) appartiennent à une organisation. Un rôle technique d'administration général est associé à chaque organisation, c'est l'équivalent du rôle super-utilisateur classique mais contraint à une organisation donnée. Les organisations sont structurées en arbre, une organisation peut contenir d'autres organisations et ses administrateurs ont tout pouvoir sur ces sous-unités administratives.

Pour chaque type d'objet on ajoute dans chaque organisation des rôles techniques d'administration de ces objets, le rôle technique d'administration général en hérite.

Error executing the **plantuml** macro (Missing partial wiki_external_filter/_macro_image with {:locale=>[:en], :formats=>[:pdf], :variants=>[], :handlers=>[:raw, :erb, :html, :builder, :ruby, :rsb]}. Searched in: * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/redmine_tags/app/views" * "/usr/share/redmine/plugins/redmine_entrouvert/app/views" * "/usr/share/redmine/plugins/plantuml/app/views" * "/usr/share/redmine/plugins/localizable/app/views" * "/usr/share/redmine/app/views")

`Vue des rôles techniques d'administration avec une unique unité administrative`

Error executing the **plantuml** macro (Missing partial wiki_external_filter/_macro_image with {:locale=>[:en], :formats=>[:pdf], :variants=>[], :handlers=>[:raw, :erb, :html, :builder, :ruby, :rsb]}. Searched in: * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/redmine_tags/app/views" * "/usr/share/redmine/plugins/redmine_entrouvert/app/views" * "/usr/share/redmine/plugins/plantuml/app/views" * "/usr/share/redmine/plugins/localizable/app/views" * "/usr/share/redmine/app/views")

`Vue sur les rôles techniques d'administration avec plusieurs unités administratives hiérarchisées`

Rôle technique d'administration des rôles

Pour chaque rôle R (technique ou pas) un rôle technique d'administration Ra est crée. Il désigne les utilisateurs ayant le droit d'ajouter ou d'enlever des membres à ce rôle ainsi que de faire hériter un autre rôle de ce même rôle. Cette dernière opération revenant à affecter tous les membres de l'autre rôle au rôle administré, il est logique de le rendre possible.

Les rôles d'administration des rôles sont particuliers en ce qu'ils contrôlent leur propre administration, i.e. l'administrateur d'un rôle pourra toujours déléguer son pouvoir à un autre, i.e. il est aussi administrateur du rôle d'administration. Ceci pour éviter une récursion infinie au niveau des rôles d'administration des rôles qui aurait eu besoin d'un rôle d'administration à leur tour. On dit que ces rôles sont 'auto-administrés'.

Pour rendre n'importe quel rôle 'auto-administré' il suffit de lui faire hériter de son rôle d'administration.

Error executing the **plantuml** macro (Missing partial wiki_external_filter/_macro_image with {:locale=>[:en], :formats=>[:pdf], :variants=>[], :handlers=>[:raw, :erb, :html, :builder, :ruby, :rsb]}. Searched in: * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/redmine_tags/app/views" * "/usr/share/redmine/plugins/redmine_entrouvert/app/views" * "/usr/share/redmine/plugins/plantuml/app/views" * "/usr/share/redmine/plugins/localizable/app/views" * "/usr/share/redmine/app/views")

Rôles techniques de gestion des accès

TODO Pour chaque object ayant besoin d'un accès on crée un rôle technique par exemple pour un service SAML w.c.s. on créera un rôle accès w.c.s, il donne accès ce service, sans lui l'IdP ne répondra pas.

À supposer que l'IdP propose de modéliser les rôles applicatifs, ils seront reproduits du coté d'authentic par des objets spécifiques, pour chacun de ces objets authentic créera un rôle d'accès du même nom. Ce rôle héritera du rôle d'accès de base.

Error executing the **plantuml** macro (Missing partial wiki_external_filter/_macro_image with {:locale=>[:en], :formats=>[:pdf], :variants=>[], :handlers=>[:raw, :erb, :html, :builder, :ruby, :rsb]}. Searched in: * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/redmine_tags/app/views" * "/usr/share/redmine/plugins/redmine_entrouvert/app/views" *

"/usr/share/redmine/plugins/plantuml/app/views" * "/usr/share/redmine/plugins/localizable/app/views" * "/usr/share/redmine/app/views")

Visibilité

Il peut se poser la question de savoir si un administrateur d'un rôle dans organisation fille doit pouvoir voir les utilisateurs ou les rôles de l'organisation parente ou d'une organisation sœur quand il administre son rôle. Pour l'instant nous écartons ce problème et le déclarons hors-scope. Dans toutes les situations où un utilisateur est amené à choisir un utilisateur, ajouter un membre à un rôle, ou un rôle sans nécessité d'une permission particulière[1] tous les utilisateurs ou rôles seront visibles.

[1] par exemple pour hériter d'un rôle il faut être administrateur de celui-ci, il ne sera donc pas possible de voir dans la liste les rôles qu'on administre pas

IHM

Le graphes des rôles est un graphe dirigé complexe qu'on essayera pas de représenter. On aura comme actuellement avec les groupes une vue alphabétique de la liste de tous les rôles, et pour chaque rôle la liste de ses membres (directs et indirects via l'héritage). On y ajoutera deux nouveaux onglets: la liste des rôles dont il hérite et la liste des rôles qui héritent de lui. Tout au plus les services pourront indiquer des relations hiérarchiques de représentation pour certains rôles techniques, ceux-ci ayant généralement des relations d'héritage simples entre eux. Ainsi il sera intéressant dans l'exemple plus haut de présenter le rôle d'accès w.c.s. comme racine d'un arbre dont les catégories en sont les branches.

Chaque organisation disposera d'une vue d'accueil présentant l'accès aux 2 objets gérés principaux les utilisateurs et les rôles. Il y sera aussi présents un bouton d'accès pour chaque sous-organisation. Les [xxx] sont des liens. [-] est un bouton de suppression, [/] un bouton de suppression grisé.

```
Agglo

  [Agglo           ]    [Ville1          ]   [Ville2] | [Ville3] | [Ville 4]
  | → utilisateurs |    | → utilisateurs |   ...
  | → (à réfléchir)|
  |                |
```

> Benj m'a dit qu'on laissait tomber cet affichage des utilisateurs par organisation (ville), c'est pas gérable quand il y a beaucoup de ville (+ de 1000 au CDG59 actuellement et beaucoup aussi dans l'AO en cours). Du coup on ne verra que les utilisateurs pour lesquels on a les droits nécessaires. Affichage sous forme de liste paginée avec une colonne indiquant l'organisation à laquelle appartient l'utilisateur.
> Idem pour les rôles.

`Accueil du manager d'authentic`

```
[Agglo] > Ville1

  [Utilisateurs] | [Rôles] | [Services SAML]

-----

  [Service enfance] [Service état civil]
```

`Accueil du manager d'authentic pour la sous-organisation ville1`

```
[Agglo] > [Ville1] > Utilisateurs

 Recherche: _________________ [Créer un nouvel utilisateur]
```

```
   Nom | Prénom | Email
1 [xxx | xxx    | xxx@xxx ]
2 [yyy | yyy    | yyy@yyy ]

  [1] [2] .. [5] [6]
```

Dans le cadre de Publik, on s'est mis d'accord sur le fait que l'accueil du /manage d'Authentic était superflu : il ne contient rien qui ne soit déjà exposé sur le portail agent (hobo/combo). Benj est ok en dépit de ce qu'il écrivait ici #7088

`Administration des utilisateurs de ville 1`

```
[Agglo] > [Ville1] > [Utilisateurs] > Utilisateur 1

 Informations | [ Rôles ]

 Nom:     xxxx_____       [Renvoyer un mot de passe]
 Prénom:  xxxx_____
 Email:   xx@xx____       [Forcer le changement du |
                          |au prochain login       ]
  [1] [2] .. [5] [6]
```

`Administration des informations personnelles de l'utilisateur 1 de ville 1`

```
[Agglo] > [Ville1] > [Utilisateurs] > Utilisateur 1

 [Informations] | Rôles

 |   | Nom
 -------------------------
 | 1 | Administrateur Ville 1 [-]
 | 2 | Administrateur de service enfance de ville 1 (rôle hérité) [/]
 | 3 | W.C.S (rôle hérité) [/]
 | 4 | W.C.S :: Service enfance [-]
 |

 Ajouter un rôle: ___________________ [Ok]
 `Vous devez être administrateur d'un rôle pour pouvoir l'ajouter`
```

(possibilité de clic sur "rôle hérité" pour avoir des infos du comment)

`Administration des rôles de l'utilisateur 1 de ville 1`

```
[Agglo] > [Ville1] > Rôles

 Recherche: _______________ [Créer un nouveau rôle]

|   | Nom                                        | Technique
-------------------------------------------------
| 1 | Administrateur de ville 1                |   Oui
| 2 | +-- Gestion des utilisateurs             |   Oui
| 3 | +-- Gestion des rôles                    |   Oui
| 4 | +-- Gestion des services SAML            |   Oui
| 5 | +-- Administration du service enfance    |   Oui <- à voir si on fait aussi apparaître les
rôles d'administration du niveau n au niveau n+1
| 6 | +-- Administration du service état civil |   Oui
```

```
| A | +-- Administration "élu"                |   Oui
| 7 | Accès à W.C.S.                          |   Oui
| 8 | +--- Service Enfance                    |   Oui
| 9 | +--- Service état civil                 |   Oui
| B | +--- Élu                                |   Oui
|10 | Élus                                    |   Non
```

(suggestion : mettre dans un onglet différent les rôles d'administration des rôles (5, 6, A))

(injonction : retirer la colonne technique)

`Administration des rôles de ville 1`

```
[Agglo] > [Ville1] > [Rôles] > Élus

 Membres | [Rôles hérités] | [Rôles héritants]

 |   | Nom | Prénom | Email    | Membre direct
 ---------------------------------------------
 | 1 | xxx | xxx    | xxx@xxx  | Oui  [-]
 | 2 | yyy | yyy    | yyy@yyy  | Non  [/]
 | 3 | zzz | zzz    | zzz@zzz  | Oui  [-]
 ....

 [1] [2] .. [3] [4]

 Ajouter un membre: ___________ [Ok]
```

(rôles hérités → rôles dont on est membre) (rôles héritants → rôles "membres de")

`Administration des membres du rôle Élus`

```
[Agglo] > [Ville1] > [Rôles] > Élus

 [Membres ] | _Rôles hérités_ | [Rôles héritants]

 |   | Nom du rôle
 -----------------------------------------
 | 1 | W.C.S :: Élu

 Ajouter un rôle hérité: ___________ [Ok]
 `Pour pouvoir ajouter un rôle hérité vous devez être administrateur de ce rôle`
```

`Administration des rôles hérités du rôle Élus`

```
[Agglo] > [Ville1] > [Rôles] > Élus

 [Membres ] | [Rôles hérités] | _Rôles héritants_

 |   | Nom du rôle                         |
 ---------------------------------------------
 | Aucun rôle héritant                     |

 Ajouter un rôle héritant: ___________ [Ok]
```

`Administration des rôles hérités du rôle Élus`

HowDoWeDoBackup

Entr'ouvert gère la sauvegarde de toutes ses installations, les sauvegardes sont quotidiennes (nocturnes) et conservées :

- quotidiennement pendant deux semaines,
- mensuelles sur 6 mois.

En plus des sauvegardes complètes à chaud des systèmes, un archivage des transactions des bases de données est réalisé, cela permet de rejouer les modifications de données et de récupérer des données à la minute. Les restaurations sont testées régulièrement. Entr'ouvert dispose d'une procédure de restauration simple, que ce soit une restauration partielle « sur site » ou totale sur une nouvelle instance.

- [[sysadmin:Procédure de récupération du backup]]

HowDoWeDoEmail

Le SPAM

- plain text c'est mieux: https://www.gkogan.co/blog/dont-design-emails/
- tester l'envoi de mail avec https://www.mail-tester.com/ (vérifie SPF, DKIM, etc.)

HowDoWeDoHotFixes

- oh un bug bloquant
- développement de la correction / relecture / push / build ok / tag recette / tests recette / etc.
- git checkout -b hotfix/whatever vLATEST
- git cherry-pick l'un ou l'autre
- git push (qui dira de faire git push --set-upstream origin hotfix/whatever)
- git push --set-upstream origin hotfix/whatever
- (attendre build jenkins)
- apt update / upgrade
    - en regardant quand même l'heure, s'il y a moyen de ne pas trop amener de coupure à un moment fréquenté

HowDoWeDoPython3Migration

On essaie au maximum d'avoir un code commun entre Python 2 et Python 3; souvent c'est possible de manière native, le reste du temps via six (pour les applications django six est dispo de base dans django.utils.six).

Éléments usuels :

- iteritems() n'existe plus en Python 3, vu les volumes de données qu'on traite, on peut partout utiliser items().
- le résultat de .values() sur un dictionnaire n'est pas de type liste, il faut donc le transformer si on veut juste en prendre un bout, genre get_wcs_services().keys()[0] → list(get_wcs_services().keys())[0].
- sort() utilise obligatoirement un paramètre key, plus moyen de faire avec cmp. ex: references.sort(lambda x, y: cmp(x[-1], y[-1])) → references.sort(key=lambda x: x[-1])
- les exceptions s'attrapent nécessairement avec la forme except Whatever as e, plus possible d'utiliser une virgule.
- on se trouvait parfois par erreur ou désir stylistique à préfixer des entiers avec un 0, genre date(2017, 01, 03), on ne peut plus faire ça.
- il est déconseillé d'utiliser file() pour ouvrir un fichier depuis Python 2.5, il faut utiliser open(), c'est désormais obligatoire.

Pour la gestion des encodages, utiliser force_text/smart_text/smart_bytes de Django, plutôt que des appels à unicode() ou str().

Via six :

La bibliothèque standard a été un petit peu rangée et certains modules ne se trouvent plus là où ils étaient; six assure la transition.

- urllib & urlparse, from django.utils.six.moves.urllib import parse as urlparse. (et/ou as urllib), ça donne accès à urlparse, urlunparse, parse_qs, quote, etc.
- from HTMLParser import HTMLParser → from django.utils.six.moves.html_parser import HTMLParser

Quand il faut nécessairement du code différent entre Python 2 et Python 3, plutôt que comparer les versions via sys.version_info il existe six.PY3 (vrai en Python 3) et six.PY2 (vrai en Python 2).

HowDoWeDoTraductions

On suit globalement ce qui a été défini dans l'équipe de tradution GNOME; documents utiles :

- https://traduc.org/gnomefr/GuideStylistique
- https://traduc.org/gnomefr/Typographie

DevGuides

- HowDoWeDoCodeReviews
- HowDoWeDoDebianPackages
- HowDoWeDoDjangoDebugToolbar
- HowDoWeDoDjangoForms
- HowDoWeDoDjangoManage
- HowDoWeDoDjangoProject
- HowDoWeDoDjangoSettings
- HowDoWeDoDjangoTenants
- HowDoWeDoDns
- HowDoWeDoEmail
- HowDoWeDoGit
- HowDoWeDoJenkins
- HowDoWeDoLDAP
- HowDoWeDoLogging
- HowDoWeDoPostgreSQL
- HowDoWeDoProvisioning
- HowDoWeDoPythonPackaging
- HowDoWeDoTests
- HowDoWeDoThemes
- HowDoWeDoSubmitFormsToWebservices
- HowDoWeDoTickets (ainsi que [[interne:Du_bon_usage_de_Redmine_dans_les_projets_clients|Du bon usage de REdmine dans les projets clients]])
- HowDoWeDoSSL par nos amis d'Evolix
- HowDoWeDoAndroidDebug
- HowDoWeDoUpgrades
- HowDoWeDoCSS

AdminGuides

- HowDoWeCreateAVM
- [[interne:Du_bon_usage_de_Redmine_dans_les_projets_internes|HowDoWeDoAdminTickets]]
- HowDoWeDoBackup
- [[sysadmin:Exécution_de_taches_sur_les_serveurs|Outil eotasks d'exécution de tâches en parallèle sur nos serveurs]]
- HowDoWeDoPublikDomainNameChange (pas sur le SaaS)
- HowDoWeDoPublikDomainNameChangeSurLeSaas

Études

- Gestion des accès

HowDoWeCreateAVM

Depuis l'interface de proxmox

Create CT

(mesclun)
Id: incrément du dernier
10xx => 192.168.43.xx
20xx => 176.31.146.xx

dns:
5.135.221.23 5.135.221.14 176.31.123.109

/!\ Sur mesclun, reloader powerdns

HowDoWeDoAndroidDebug

Pour debugguer un environnement Publik local avec Chrome Android ("remote debugging").
Il ne s'agit pas ici de debugger des application Android, mais simplement d'afficher une URL sur son téléphone et de voir tout ce qui se passe dans le debugguer de son bureau.

Pré-requis

Installez Chrome (version google nécessaire pour "inspecter" un onglet sous android chrome)

```
google-chrome --remote-debugging-port=9222
```

TODO trouver comment inspecter une page android-chrome avec chromium sous debian

(note de Fred : simplement utiliser chromium également sur le téléphone)

La documentation complète est ici : https://developers.google.com/web/tools/chrome-devtools/remote-debugging/

Configurez votre serveur local

Pour développer localement sur un module de Publik, utilisez publik-devinst en suivant cette documentation

Vous aurez toutefois besoin d'adapter cette installation pour que les DNS de votre installation locale soient aussi accessibles depuis le monde extérieur.

- configurez votre IP interne pour correspondre à votre dns de développement en allant sur https://www.entrouvert.org/nsupdate?name=dev&ip=192.168.x.y
  - cela créera une redirection pour *.dev-[login-eo].ddns.entrouvert.org
  - voir la documentation Whitelist EO

- Adaptez les instructions pour signer vos certificats SSL avec la CA d' EO en suivant le wiki sur publik-devinst * Adaptez aussi les instructions pour configurer votre DNS local afin de diriger sur localhost * "Adaptez le déploiement des nouveaux tenants pour avoir des tenants déployés avec des noms du style *.dev-[login-eo].ddns.entrouvert.org

Configurez le téléphone Android de développement.

- installez android-tools-abd sur votre système pour communiqer avec votre Android en debug USB par le biais de la ligne de commande.

```
sudo apt install android-tools-adb
```

- Téléchargez et ajoutez l'autorité CA EO
- téléchargez sur votre laptop ce fichier .crt en format DER
- transférez-le sur la mémoire :

```
adb push entrouvert-ca-der.crt /mnt/sdcard/Download/
```

- installez le certificat dans votre système Android en suivant ce lien
  - à la validation de l'autorité, choisissez le type (étape 8) : VPN or apps

Debugguez !

- Ouvrez les devtools de chromium, et allez à l'onglet remote debugging et envoyez l'URL que vous voulez debugguer, par exemple https://user-combo.dev-[login-eo].ddns.entrouvert.org/

HowDoWeDoCodeReviews

Parce que ça améliore la qualité du code produit, que ça permet un partage de la connaissance du code, on fait des relectures des patchs avant d'envoyer ceux-ci dans les dépôts. Pour citer "Code reviews: from nitpicking to cooperation" repris dans les ressources en bas de page :

> Code review and peer review are great methods for cooperation aiming at:
>
> - Ensuring that the code works as intended
> - Ensuring that the task was fully accomplished and no detail left out
> - Ensuring that no security issues have been introduced
> - Making sure the code fits the practices of the team and is understandable and maintainable by others
> - Sharing insights, transferring knowledge between code author and code reviewer
> - Helping the code author to learn to write better code

Toutes les applications, tous les modules sont concernés, que ça soit mail2redmine ou w.c.s., et de la même manière, tout le monde est concerné. Comme à tout il peut y avoir des exceptions, elles sont listées en bas de cette page.

Timing et attentes

Un auteur n'a pas envie d'attendre des jours que quelqu'un jette un œil à son code; il est souvent utile de préférer une réponse rapide mais incomplète, concentrée sur les points majeurs relevés d'une première lecture, plutôt qu'attendre le moment permettant la relecture intégrale et exhaustive. À ce sujet ça peut sembler ridicule de pointer des problèmes de style dès le premier moment mais c'est surtout ridicule parce que de tels problèmes ne devraient pas arriver avec un environnement de développement adéquat, qui indente de manière uniforme par exemple. En pratique l'utilisation de black est mise en place dans Chrono, pourrait être généralisée une fois intégrée dans jenkins etc.

Comme les relectures un autre aspect important du développement concerne les tests, de manière graduée une règle peut être qu'un patch doit être couvert par des tests autant que le reste du code du module (cf jenkins pour prendre connaissance de la couverture de code). Avoir les commits poussés dans des branches "wip" permet à jenkins d'assurer l'exécution et la visibilité des tests.

À noter qu'on peut bien sûr également partager des patchs en amont, pour interroger sur une approche, on ne fait pas de test-driven development.

Esprit de relecture

L'esprit doit être positif etc. L'article "Code reviews: from nitpicking to cooperation" en ressource pointe bien les risques à exiger des commits "parfaits".

Méthodes particulières

- patch CSS: avoir un screenshot avant (par l'auteur du ticket) et après (par le responsable du ticket) permettant une relecture facile.

Exceptions

- tabellio.*, themis.*, pfwbged.* et les autres modules Plone; on doit faire là avec un historique de code et technos mal partagé, Frédéric y est tout seul :/

Ressources

- https://curlybracket.net/2020/08/24/nitpicking.html

Files

patch.png	78.3 KB	06 Apr 2018	Frédéric Péters

HowDoWeDoCSS

Conseils et recommandations dans la rédaction ou refactorisation de code CSS.

Commentaires

La compilation sass du code source n'effectue pas de compression du code CSS compilé. Il est donc déconseillé d'utiliser les blocs commentaires CSS /* commentaire */.
À la place, utiliser des commentaires Sass // commentaire.

Plusieurs fichiers

Évitez les fichiers CSS à rallonge.
Il est judicieux de scinder des fragments de code distincts dans leurs propres fichiers, qui sont concaténés lors d'une étape de construction.
Isoler chaque composant/bloc dans un fichier propre permet de les faire évoluer distinctement des autres sans craindre les conflits ou d'éventuelles régressions sur les autres.

Tous les styles d'un composant seront donc de préférence définis dans son fichier.
Il faut éviter de styler plusieurs éléments provenant de plusieurs composants / blocs au sein d'une même déclaration.

Ne pas faire :

```
.composant-1 h2,
.bloc-2 h2,
.composant-3 h3 {
    text-transform: uppercase
}
```

Sélecteurs CSS

Liste des bonnes pratiques pour sélectionner et nommer un élément en CSS.

Éviter les sélecteurs d'id

Les sélecteurs par `#id` ont une spécificité trop lourde dans le calcul de résolution de la cascade, empêchant ainsi l'écriture efficace d'un code modulaire.

Éviter les sélecteurs de tag

Utiliser le sélecteur de tag doit être considéré comme une **très mauvaise pratique**.
Cela revient à définir le style d'un bloc en rapport à sa sémantique et rend impossible la mise à jour de l'HTML.

Exemple et conséquence de cette mauvaise pratique. **Ne pas faire :**

```
div.composant {
    background-color: gray;
}
```

Quand, dans le fichier *template* du composant, il s'avérera judicieux de remplacer la balise `div` non sémantique par une balise `section`, le style ne s'appliquera plus. Ce sélecteur empêche la mise à jour du *template*.

No DOM

Cette règle suit logiquement les recommandations précédentes.
Évitez les sélecteurs descendants par tag ou #id, **Ne pas faire :**

```
div.composant > ul > li {
    padding-left: 1em;
}
```

Dans cet exemple, les modifications suivantes du *template* HTML généreront des régressions de styles :

• La modification des balises div, ul ou li par des choix sémantiques différents.

• L'ajout des balises intermédiaires, comme ajouter une balise `nav` entre `div` et `ul`.

Ne pas ajouter d'informations relatives au DOM dans vos sélecteurs.

First class

Il est donc préférable de cibler les éléments uniquement par leur class.
Pour permettre une sélection sécurisée d'un élément par une class unique, une convention de nommage excluant les conflits est nécessaire.

Convention `block--element`

Le modèle `.app-block--element` est préconisé pour l'écriture de tout nouveau composant ou bloc.

- `app` : préfixe optionnel mais utile pour indiquer l'application générant le bloc.
- `block`: identifiant unique de la racine du bloc ou composant.
- `element`: élément/enfant d'un bloc

block

Règles à suivre pour la définition du nom de class du bloc.

- La class d'un bloc doit être unique.
- La class d'un bloc peut se composer du préfixe de l'application qui le construit
- Si préfixe `app`, il est séparé de l'identifiant par un trait d'union (abusivement appelé tiret du 6): `.app-block`.
- L'identifiant peut être composé de plusieurs parties séparées par 1 trait d'union: `.app-composant-extra`, `latest-updated-pages-cell`. Pour des raisons de lisibilité, l'utilisation de traits d'unions est recommandé:
  - `latestupdatedpagescell` NON
  - `latest-updated-pages-cell` OUI
- L'utilisation du CamelCase n'est autorisé **que dans le cas** où l'identifiant du composant nécessiterait plus d'un trait d'union: `.wcs-cell-currentDrafts`

element

- La class d'un élément est composé de l'identifiant de son bloc en préfixe, suivi de l'identifiant de l'élément, séparé par 2 tirets `.block--element`
- L'identifiant de l'élément peut-être composé d'un trait d'union: `.bloc--sub-title`, préférable à l'utilisation du camelCase.
- On ne peut pas coller 2 éléments au nom d'un bloc pour définir un sous element: `.nav--sub-nav--link` n'est pas autorisé. Dans ce cas, définir un nouveau bloc `.sub-nav`.

Dictionnaire d'identifiants d'éléments

- --wrapper
- --list
- --item
- --title
- --body
- --link
- --metas
- --img

Élément ou sous-bloc ?

Lorsqu'un composant présente une arborescence profonde, comme c'est souvent le cas avec les listes, il est préférable d'imbriquer les blocs plutôt que de définir des éléments trop complexes.

```
.main-nav {}
.main-nav--list {}
.main-nav--item {}
.main-nav--link {}

.sub-main-nav {}
.sub-main-nav--list {}
.sub-main-nav--item {}
.sub-main-nav--link {}
```

Exemple

Exemple d'un fichier scss `wcs/_steps.scss` avec le composant 'étapes d'une demande'.

Il est composé de 2 blocs

- `.wcs-steps` identifiant racine du composant
- `.wcs-step` identifiant du composant

```
.wcs-steps              { /* styles du bloc */ }
.wcs-steps--list        { /* styles pour la liste (ol) */ }

.wcs-step               { /* styles pour l'item (li) */ }
.wcs-step--marker       { /* styles pour le marker (abbr) */ }
.wcs-step--marker-nb    { /* styles pour le num du marker (span) */ }
.wcs-step--marker-total { /* styles pour le num total du marker (span) */ }
.wcs-step--label        { /* styles pour le label (p) */ }
```

Avec le *nesting SASS*, il est possible de ne pas répéter l'identifiant du bloc dans les class des éléments tout en représentant graphiquement l'arborescence DOM des balises du *template*.

```
.wcs-step {
    /* styles pour l'item (li) */
    &--marker {
        /* styles pour le marker (abbr) */
        &-nb {
            /* styles pour le num du marker (span) */
        }
        &-total {
            /* styles pour le num total du marker (span) */
        }
    }
    &--label {
        /* styles pour le label (p) */
    }
}
```

HowDoWeDoDebianPackages

Pour un backport de paquet qui existe déjà dans Debian, on peut utiliser dgit :

```
 $ dgit clone django-haystack
 $ cd django-haystack
 $ # si debian/patches/series existe, le retirer, parce que eobuilder va recréer le .orig.tar.gz avec les patc
hs déjà appliqués
 $ git rm -f debian/patches/series && git commit debian/patches/series
 $ git remote add eo git@git.entrouvert.org:debian/django-haystack
 $ git push -u eo dgit/sid:master
```

Puis lancer https://jenkins.entrouvert.org/job/debs/.

Paquets python

- utiliser pybuild
- déclarer le nom de la distribution python
- désactiver les tests
- pour les paquets d'applications Django, recopier manage.py du paquet python vers le paquet démon

debian/rules

```
#!/usr/bin/make -f
# -*- makefile -*-

export PYBUILD_NAME=combo <-- ici nom de la distribution déclaré dans setup.py ou pyproject.ml, ou plutôt suff
ix qui suivra python(3)- dans le nom du paquet, j'ai un doute ici
export PYBUILD_DISABLE=test

%:
    dh $@ --with python3 --buildsystem=pybuild <-- rajouter python2 si besoin, systemd si brique

override_dh_install: <-- uniquement pour les briques
    dh_install
    mv $(CURDIR)/debian/python3-<project>/usr/bin/manage.py $(CURDIR)/debian/<project>/usr/lib/<project>/manag
e.py
```

debian/python(3)-x.install

Inutile.

debian/project.install pour les briques (applications Django)

On ne doit trouver que ces fichiers (.init, .service, .cron, etc.. installé automatiquement)

```
debian/combo-manage       /usr/bin
debian/settings.py        /etc/combo
debian/uwsgi.ini          /etc/combo
debian/debian_config.py   /usr/lib/combo
```

Le manage surchargé, le settings de base, le uwsgi de base et le debian_config.py de base.

Notes eoday

- debian-$dist/ dans master
- pas de job *-deb dans jenkins (un job unique)
- génération d'un changelog from scratch
- génération VERSION depuis tag ou prendre 0.0

ex de fichier changelog à générer :

```
wcs (1.12.12.5.g46a0062-1~eob70+1)
```

```
  * (46a0062) tests: check forgotten page where user can set password (#6506)
  * (edbd295) hobos: change command to only deploy one instance, given as arg (#6486)
  * (0a58b8f) hobos: rename command to hobo_deploy, to match others (#6486)
  * (d0e2120) hobos: read json from a file if a filename is given on the command line (#6486)
  * (377911a) email: do starttls if advertised by the server (#6453)

wcs (1.12.12-0) $dist; urgency=low

  * (8df6312) release 1.12.12
  * (c631b0a) ctl: fix display of list of commands
  * (b199f1d) update french translation

 -- eobuilder <eobuilder@entrouvert.com>   Date...

wcs (1.12.11-0) $dist; urgency=low

  * (a563be0) release 1.12.11
  * (e54d2eb) form: fix handling of date in iso format in French-configured sites (#6390)
  * (1505495) use new password entry widget for register page (#5805)
  * (4a91c26) form: turn file type check into a hard check (#6134)
  * ...

 -- eobuilder <eobuilder@entrouvert.com>   Date...

...
```

Comment modifier la configuration d'un paquet debian d'un de nos projets ?

Dans la dépôt git du projet, les fichiers qui sont "usuellement" édités sont à la racine du dépôt dans le dossier debian/

- Exemple de fichier debian/control:

```
Package: publik-base-theme
Architecture: all
Depends: ${shlibs:Depends}, ${misc:Depends}, python-xstatic-abrilfatface
Conflicts: python-authentic2 (< 2.1.20.742.gb6ee096-0)
Breaks: combo (< 0.7.1)
Description: Publik Base Theme
```

- Exemple de fichier debian/publik-base-theme.dirs:

```
/etc/combo/settings.d
```

- Exemple de fichier debian/publik-base-theme.install:

```
debian/fonts.py        /etc/combo/settings.d
```

Comment tester localement l'empaquetage debian ?

Pour ne pas avoir à pousser sur master et attendre de voir la tâche jenkins de notre paquet (par ex. https://jenkins.entrouvert.org/job/publik-base-theme-deb/), on peut localement tester ça sur sa debian avec :

```
sudo apt install devscripts
cd monprojet
debuild -uc -us -b
```

HowDoWeDoDjangoDebugToolbar

Préparation

Aussi appelé djdt.

Depuis la version 1.6 l'installation automatique a été retiré il faut explicitement ajouter le code suivant dans le urls.py principal de la brique où on souhaite utiliser djdt:

```
if settings.DEBUG and 'debug_toolbar' in settings.INSTALLED_APPS:
    import debug_toolbar
    urlpatterns = [
        url(r'^__debug__/', include(debug_toolbar.urls)),
    ] + urlpatterns
```

Ensuite poser le fichier 10djdt.py suivant dans le répertoire settings.d de la brique:

```
DEBUG = True

# placer le middleware djdt après le middleware qui gère ForwardedFor, sinon on ne verra pas la bonne IP
MIDDLEWARE_CLASSES = MIDDLEWARE_CLASSES[:1] + (
    'debug_toolbar.middleware.DebugToolbarMiddleware',) + MIDDLEWARE_CLASSES[1:]

INSTALLED_APPS += ('debug_toolbar',)

INTERNAL_IPS = ['176.31.123.109']  # IP du VPN
```

Debian jessie

Le paquet de jessie n'est pas compatible Django 1.8 mais le paquet de stretch l'est et s'installe sans problème sur jessie pourvu qu'on installe aussi le paquet python-sqlparse de stretch.

- https://packages.debian.org/fr/stretch/python-sqlparse
- https://packages.debian.org/fr/stretch/python-django-debug-toolbar

HowDoWeDoDjangoForms

- have a <project>/form.html template with content

```
{{ form.as_p }}
```

- define a form.html in publik-base-theme, inherit from it in each brick's templates

HowDoWeDoDjangoManage

On laisse le fichier manage.py.

Dans le packaging debian on le met dans /usr/lib/$project/manage.py.

On crée un script /usr/bin/package-manage qui pose la variable d'environnement pour les settings debian et appelle le script manage, on oublie les jeux avec sudo/su au mieux on émet un message d'erreur quand l'utilisateur n'est pas bon.

HowDoWeDoDjangoProject

Pour éviter de répéter tout le temps les mêmes erreurs c'est intéressant d'avoir un projet template qui rassemble les bonnes pratiques sous forme de code et pas juste d'un wiki:

https://dev.entrouvert.org/projects/eo-django-project-template

On y met les bonnes recette de packaging python et debian mono et multi tenant.
Les bons settings, les bonnes librairies tierces qu'on voudrait voir partout (django-compressor?). La bonne intégration dans jenkins (tox.ini ? jenkins.sh ?).

HowDoWeDoDjangoSettings

Application simple

Le fichier settings.py est celui créé par Django (lors du startproject), on y ajoute la possibilité de charger des paramètres supplémentaires en ajoutant ces lignes à la fin (avec FOOBAR remplacé par le nom du projet) :

```
local_settings_file = os.environ.get('FOOBAR_SETTINGS_FILE',
        os.path.join(os.path.dirname(__file__), 'local_settings.py'))
if os.path.exists(local_settings_file):
    execfile(local_settings_file)
```

Commentaires

Par rapport à la forme from local_settings import * précédemment pratiquée, ça a l'avantage de donner un accès aux différentes variables déjà définies, pour par exemple permettre au fichier donné d'altérer INSTALLED_APPS.

Un inconvénient est que le fichier n'est pas surveillé par django, les modifications qui y sont faites ne sont pas automatiquement prises en compte, un redémarrage (ou la modification d'un autre fichier) est nécessaire.

Application multitenant

Le multitenant s'obtient pour une application donnée en utilisant le package python-hobo (à ne pas confondre avec l'application hobo). Pour devenir multitenant, une application doit entre autres modifier ses settings. Les modifications requises sont encapsulées dans un fichier de configuration debian_config_common.py que l'installation du package python-hobo positionne dans /usr/lib/hobo/.
Considérons une des briques applicatives de Publik, appelons là 'foobar'. La succession de chargement des settings se produit commme suit:

- l'application charge son fichier de config par défaut settings.py que l'on trouve dans les sources à la racine du package 'foobar' : /usr/lib/pythonX.X/dist-packages/foobar
- foobar.settings.py charge le fichier pointé par la variable d'environnement FOOBAR_SETTINGS_FILE comme décrit dans le cas d'une application simple, ce fichier a généralement pour nom debian_config.py et est positionné dans /usr/lib/foobar/
- debian_config.py charge /usr/lib/hobo/debian_config_common.py

L'essentiel de la configuration est chargé à ce moment là. Cependant, si ils sont présents, des fichiers additionnels de configuration seront encore chargés:

- /etc/foobar/settings.py

Et les éventuels

- /etc/foobar/settings.d/*.py

HowDoWeDoDjangoTenants

Individuellement les projets n'ont pas à se soucier de multitenant, c'est totalement pris en charge par de la configuration.

Voir http://git.entrouvert.org/hobo.git/tree/debian/debian_config_common.py?h=master (à partir de # multitenant adaptations, ligne 206)

Exemple de fichier /usr/lib/<project>/debian_config.py appelé à la fin du settings.py du projet :

```
# /usr/lib/<project>/debian_config.py
# appelé en execfile() en toute fin des settings.py de <project>

PROJECT_NAME = 'foobar'

# si nécessaire:
#INSTALLED_APPS += ('mellon',)    # SAML2 authentication

execfile('/usr/lib/hobo/django_config_common.py')    # cf http://git.entrouvert.org/hobo.git/tree/debian/debian
_config_common.py?h=debian

# en cas de besoin supplémentaire au niveau de la gestion des settings.py
# par tenant pour <project>. Par exemple pour Authentic:
TENANT_SETTINGS_MIDDLEWARE_LOADERS = +(
        'hobo.multitenant.settings_loaders.Authentic',
        # 'hobo.multitenant.settings_loaders.SettingsPy',
)

execfile('/etc/%s/settings.py' % PROJECT_NAME)
```

Utilisation

Création d'une base de données (avec comme propriétaire l'utilisateur faisant tourner le code de l'appli)

```
# su - postgres
$ createdb -O user foobar
```

Création d'un tenant

```
$ ./manage.py create_tenant foobar.example.net
[...]
$ ls /var/tmp/foobar/tenants/foobar.example.net/
media  static  templates
```

Appel des commandes de management

```
$ manage.py tenant_command createsuperuser -d foobar.example.net
Username (leave blank...)
```

Visite avec postgresql

```
$ psql foobar
foobar=> set schema 'foobar_example_net';
SET
foobar=> select username from auth_user;
 username
----------
 fred
(1 ligne)
```

```
...
```

Fonctionnement

tenant_schema, "schémas" dans postgresql, + de notre côté des fichiers dans le filesystem, /var/lib/xxx/tenants/yyy.

Lancer des commandes

Commande unique:

```
package-manage tenant_command command -d domain ...
```

Commande multiple (pour les cron):

```
package-manage tenant_command command --all-tenants domain ...
```

Renommage des tenants

Renommage d'un site wcs (note: rien à voir avec les tenants Django)

- déclarer le domaine et le vhost nginx
- se rendre dans le backoffice de wcs /backoffice/settings/identification/idp/sp et /backoffice/settings/sitename et modifier les urls avec le nouveau nom.
- renommer le répértoire du tenant
- mettre à jour les méta-données dans authentic à partir de la nouvelle url des métadonnées
- mettre à jour l'url de base dans le serveur hobo (dans le shell django)
- dans l'ancien vhost nginx mettre en place la redirection vers le nouveau nom.

Migrations Django

Il faut utiliser une commande dédiée pour gérer les migrations sur les tenants.

Revenir à la migration d'avant

```
$ passerelle-manage migrate_schemas iparapheur 0005 -v2
$ passerelle-manage migrate_schemas iparapheur 0005 -v2 --fake  # uniquement en base
```

Ajouter une migration

```
$ passerelle-manage makemigrations
$ git add passerelle/contrib/iparapheur/migrations/0006_iparapheur_wsdl_endpoint_location.py
$ passerelle-manage migrate_schemas
```

HowDoWeDoDns

Pour les déploiements de Publik on a besoin d'une série de noms de domaine et il faut les demander au client; là on pourrait 1) leur pointer l'adresse IP du service pour faire un enregistrement de type A mais ça nous bloquerait ensuite si jamais on devait changer celle-ci (par exemple si on met de l'ip failover en amont, si on change de presta, etc.), 2) on pourrait leur demander de faire un CNAME vers le service, genre statis.example.net CNAME bi.entrouvert.org. mais si jamais on se trouve devoir ou décider de distribuer les clients différemments, ça va nous bloquer aussi.

La solution adoptée est simplement de faire du nom de domaine un CNAME vers "nom de domaine".entrouvert.org, ainsi on a client par client, service par service, la main.

Exemple :

```
statistiques.nancy.fr      CNAME statistiques.nancy.fr.entrouvert.org.
```

https://perso.entrouvert.org/~fred/dns.html existe pour ridiculement faire ça (ajouter .entrouvert.org).

Ensuite, dans la configuration de notre côté (sur leucas), on fait le CNAME approprié. (dans cet exemple de statistiques.nancy.fr.entrouvert.org vers bi.entrouvert.org)

Zone *.dev.entrouvert.org

Il y a powerdns et du code et ça marche sans intervention manuelle.

HowDoWeDoGit

- /!\ **On ne fait JAMAIS git push -f sur la branche main.** /!\

Travailler dans une branche dont le nom suit le convention de nommage **wip/XXXXX-mini-description**, où XXXXX est le numéro de ticket où le développement est suivi.

```
git checkout -b wip/XXXXX-mini-description
```

Une fois les commit faits dans la branche, pousser pour permette à Jenkins d'exécuter les tests :

```
git push origin wip/XXXXX-mini-description
```

Les tests seront exécutés par Jenkins et un mail vous sera envoyé en cas d'erreur.

- dans ~/.gitconfig:

```
[color]
    branch = auto
    diff = auto
    interactive = auto
    status = auto
[pull]
    rebase = true
```

- séparer les patchs de traduction des patchs de code
- pareil, ne pas mêler des modifications de style (genre réindentation ou autre) dans un patch ajoutant une fonctionnalité/corrigeant u nbug.

- git log
- git show (avec la coloration syntaxique activée plus haut, ça permet de voir les espaces qui trainent, par exemple)
- git add -p

- travail dans une branche, git rebase main avant de merger
  - git cherry-pick aussi, peut-être

- travail dans main, git rebase -i origin/main, taper les commits qu'on veut en premier
  - git push origin hash:main
  - ou second git rebase -i, effacer les commits en trop, noter le hash
    - git push; git merge hash

- retrouver l'état d'une branche avant un rebase, un merge, un git --amend: git reflog , une fois retrouvé le bon commit on y retourne de façon non destructive par git reset <commit> ou destructive (on perd les différences) par git reset --hard <commit>

- revenir sur le le dernier commit pour le découper (git reset HEAD^)

- ajouter les modifications en cours au dernier commit (git commit -av --amend)

- réorganiser manuellement les n (ici 5) derniers commits (git rebase -i HEAD~~~~~, ou git rebase -i HEAD~5)

(?) http://blogs.gnome.org/bastian/2015/02/12/a-guide-to-git-in-gnome-for-the-simple-minded/

- pour obtenir 8 lignes de contexte dans les patchs, ajout à ~/.gitconfig:

```
[diff]
  context = 8
```

- Pour merger des fichiers .po sans se faire chier: https://gist.github.com/mezis/1605647

- Gérer des release de branche stable, i.e. x.y.z avec z > 0:
  - on démarrer un branche release-x.y.z en partant du tag vx.y.(z-1)
  - on cherry-pick les choses de main qu'on veut garder, on commit les choses qui sont spécifique à cette release mais qu'on ne veut pas dans main (souvent des contournements temporaires de bugs, on peut aussi commiter sur release et espérer merge dans main mais c'est plus

aléatoire)
    - quand c'est fini:
        - mettre à jour le changelog
        - git tag -a -m 'x.y.z' vx.y.z release-x.y.z
        - git checkout main
        - git merge -s ours vx.y.z (si on a fait des cherry-pick si on a commmité directement, faudra faire un vrai merge, sans le '-s ours')
        - git branch -r release-x.y.z
        - git push origin :release-x.y.z (si on l'avait poussé dans le dépôt central)

- Sur la forme des messages de commit, http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html
    - Modulo la première ligne il y est écrit "max 50 caractères", ça peut être 80. Et pas nécessairement de majuscule au début.
    - Et inclure une référence au ticket, façon (#12345) par exemple : doc: explain how we are supposed to commit (#12345)
        - Le message du commit doit être en anglais, au présent.

```
Describe your changes in imperative mood, e.g. "make xyzzy do frotz"
instead of "[This patch] makes xyzzy do frotz" or "[I] changed xyzzy
to do frotz", as if you are giving orders to the codebase to change
its behaviour.
 -- http://git.kernel.org/cgit/git/git.git/tree/Documentation/SubmittingPatches
```

- Certains dépôts présentent des commits annotés (voir git-notes). Ces notes ne sont pas tirées par défaut. Pour ce faire il suffit d'ajouter la référence au remote correspondant, dans le fichier de configuration git du dépôt. Par exemple, dans l'entrée [remote "origin"], y ajouter :

```
    fetch = +refs/notes/*:refs/notes/*
```

Créer sur git

man gitolite
https://git-scm.com/book/en/v1/Git-on-the-Server-Gitolite

- cloner le dépôt de la config gitolite

```
git clone git@git.entrouvert.org:gitolite-admin
```

Créer un utilisateur

- copier la clé publique de l'utilisateur à rajouter dans gitolite-admin/keydir

```
cp ~/.ssh/id_rsa.pub $USER
```

Créer un dépôt

- éditer gitolite-admin/conf/gitolite.conf et y ajouter (essayer de conserver un ordre alphabétique entre les projets):

```
repo    monbeau-project
        RW+     = @eo
        R       = @tools
...
monbeau-projet = "Un super projet qu'il est bien"
```

La ligne de description est à ajouter en fin de fichier (toujours en respectant un ordre alphabétique)

Porcelaine git pour notre workflow

J'ai créé ça :

https://pypi.org/project/git-redmine/

C'est sympatoche:

- pip install --user git-redmine
- poser ça dans ~/.config/git/config:

```
[redmine]
url·=·https://dev.entrouvert.org/
key·=·<apikey à récupérer sur redmine sur la page de votre utilisateur>
```

- git redmine project set <redmine-project-slug> : associe le dépôt git à un projet
- git redmine issue new : ouvre un $EDITOR pour décrire un nouveau ticket, obtient un numéro et crée la branche wip/XXXX-description pour commencer à bosser
- git redmine issue take XXXX : prendre un ticket existant pour bosser
- git redmine rebase : fais un pull sur main puis vérifie que la branche courante rebase sur main
- git redmine issue submit : fait un rebase comme juste avant, puis si ça passe poser les patchs dans le ticket en passant le ticket en Solution Proposée (ça demande si on veut changer le status et s'affecter le ticket)
- git redmine merge-and-push : pull un main récent, vérifie qu'un rebase de la branche est possible, puis merge la branche, supprime une éventuelle branche distante si créée pour jenkins, pousse sur origin/main puis demande s'il faut supprimer la branche locale

Jobs pipeline

Deux jobs jenkins par projet : un job projectname et un job projectname-wip.
Les deux jobs utilisent le même Jenkinsfile, placé à la racine du dépôt git du projet projectname.

Le Jenkinsfile :

- utilise tox pour exécuter les tests (qui lui même utilise pytest et pylint)
- build les paquets debian (si il est appelé par le job projectname)
- publie les rapports de tests, coverage, pylint
- notifie les résultats de build par mail (par https://wiki.jenkins.io/display/JENKINS/Mailer)

Une libraire est utilisé pour éviter la redondance entre les Jenkinsfile des différents projets (http://git.entrouvert.org/jenkins-lib.git/).

Documentation jenkins pipeline : https://jenkins.io/doc/book/pipeline/
Documentation jenkins shared library : https://jenkins.io/doc/book/pipeline/shared-libraries/

Job projectname

- job de type pipeline
- ne s'exécute que sur la branche master (déclenchement automatique par scrutation de changements sur le dépôt git toutes les 5 minutes, et à défaut de changements une fois par jour)
- exécute les tests, ne build pas les paquets debian
- notifie les résultats de build par mail à une liste de diffusion : admin+jenkins-projectname@entrouvert.com

Job projectname-wip

- job de type pipeline multibranch
- s'exécute que sur les branches wip/* (déclenchement automatique par scrutation de changements sur le dépôt git toutes les 5 minutes)
- exécute les tests et build les paquets debian dans la foulée
- notifie les résultats de build par mail à l'auteur du dernier commit de la branche

Jobs classiques (déprécié)

- créer un jenkins.sh dans chaque projet
- utiliser uniquement tox pour organiser les tests, pylint, génération de la doc, etc..
- si plusieurs suites de tests utiliser merge-junit.sh et merge-coverage.sh du projet authentic2
- ne pas activer l'option "envoyer des emails aux personnes qui ont cassé le build" pour les projets tiers
- configurer un build par scrutation du git + un build périodique (@daily, utile pour détecter au plus tôt des problèmes avec des mises à jour de dépendance)
- ne pas configurer les jobs "-deb" avec une scrutation de gir, les configurer pour constuire après le build du projet
- pour tester les branches de développement il suffit déclarer un "branch specifier" avec un wildcard, ex.: wip/*, attention si cela affecte le statut je job quelque soit la branche, il vaut donc mieux le faire sur un job spécifique (nommé par exemple <project>-wip-branches)

HowDoWeDoLDAP

Configurer authentic pour le LDAP LE

- Ajouter le code suivant dans /etc/authentic2-multitenant/config.py (ou /etc/authentic2/config.py pour un authentic monotenant):

```
LDAP_AUTH_SETTINGS=[
        {
            "url": "ldaps://ldap.libre-entreprise.org/",
            "basedn": "o=entrouvert,ou=companies,o=libre-entreprise",
            "user_filter": "uid=%s",
            "username_template": "{uid[0]}@{realm}",
            "groupsu": ["cn=ldapadmins,ou=groups,o=entrouvert,ou=companies,o=libre-entreprise"],
            "groupstaff": ["cn=ldapadmins,ou=groups,o=entrouvert,ou=companies,o=libre-entreprise"],
            "group_filter": "(&(uniqueMember={user_dn})(objectClass=legroup))",
            "create_group": True,
            "attributes": [ "uid" ],
            "set_mandatory_groups": ["LDAP Entrouvert"]
         }
]
```

- Installer le paquet ee-ca qui contient le certificat racine de l'autorité de certification Easter-Eggs qui a produit le certificat du serveur LDAP LE
- Pour plus de sûreté on peut aussi ajouter notre copie du serveur ldaps://cresson.entrouvert.org et notre AC avec le paquet ca-certificates-entrouvert

BABA de la connexion à un LDAP

- Demander le root DSE, i.e., la description du serveur par lui même

```
ldapsearch -H ldap[s]://hostname[:port]/ -b '' -s base -x +
```

On trouvera la liste des base DN dans l'attribut namingContexts .

- Pour voir si la connexion est bloquée par des certificats tenté un ldapsearch avec LDAPTLS_REQCERT=never

Configuration

- user_filter : filtre LDAP permettant de trouver un utilisateur. Doit obligatoirement contenir au moins un **%s** qui sera remplacé par l'identifiant de l'utilisateur. Sans **%s** la tentative d'authentification est interrompue.

HowDoWeDoLogging

La configuration globale des logs se fera via des variables d'environnement positionnée au niveau de la configuration de systemd dans /etc/systemd/system.conf via la directive DefaultEnvironment. Sur un système en marche on fera ensuite systemctl daemon-reexec pour s'assurer du rechargement de la configuration, aussi on relancera les démons concernés.

Le niveau NOTICE n'existe pas en Python, on l'ajoutera et on lui donnera comme valeur 25.

```
DefaultEnvironment="SENTRY_DSN=https://a2009....ouvert.org/1" "GRAYLOG_URL=ufo.entrouvert.org:12203"
```

mettre aussi dans /etc/environment pour que l'environnement utilisateur et système soit équivalent:

```
SENTRY_DSN=https://a20092f7b3da484abf8fc0a4ee2f7202:c9fab41dcde24b04a30ac01ffd3c2457@sentry.entrouvert.org/1
```

Sentry

- Installer python-raven et python-requests
- Pour les logiciels basés sur hobo raven est configuré via la variable d'environnement SENTRY_DSN #10293 #10294
- Pour les autres:

```
INSTALLED_APPS += ('raven.contrib.django.raven_compat', )

RAVEN_CONFIG = {
    'dsn': 'https://a20092f7b3da484abf8fc0a4ee2f7202:c9fab41dcde24b04a30ac01ffd3c2457@sentry.entrouvert.org/1',
}
```

- Sur wheezy où python-raven est trop vieux, ajouter ?verify_ssl=0

- Dév: https://a20092f7b3da484abf8fc0a4ee2f7202:c9fab41dcde24b04a30ac01ffd3c2457@sentry.entrouvert.org/1
- Recette : https://b688adc1efda4a2e80b3c903974200d3:5aa2259879e24b82b70cbbed5d05986a@sentry.entrouvert.org/2
- Prod : https://a128e08c0c9b45a1b080d8f310c2a964:1c5bdf42a28f479696e2a703717f2c32@sentry.entrouvert.org/3

Graylog

- TODO

Dans les scripts

- Créer un logger global et appeler basicConfig pour envoyer les erreurs sur sys.stderr:

```
import logging
logging.basicConfig(format='%(asctime)-15s $(levelname)s %(message)s', level=logging.INFO/WARNING/ERROR) # définir le niveau en fonction d'une option --verbose par exemple
logger = logging.getLogger('monscript')
logger.info('python was here')
```

Configuration des loggers

Je commencerai avec quelques principes:

- par défaut (le settings.py du projet django) en console, et uniquement les domaine project_name et django au niveau INFO.
- dans le debian_settings.py du packaging idem mais vers syslog
- toujours disable_existing_logger à True sinon on se retrouve avec les logs gunicorn dans /var/log/syslog ou la console
- ne JAMAIS au grand JAMAIS faire un logging.getLogger() dans un contexte global (ex.: logger = logging.getLogger(__name__) si le module s'exécute avant le chargement des settings; le logger sera désactivé avant même d'avoir pu servir. On demande un logger quand on en a besoin pas avant.

Questions:

- est-ce qu'on remet des trucs magiques qui mettent le niveau des loggers à DEBUG lorsque DEBUG est à True ? C'est pratique mais c'est un

retour en arrière parce que cela demande de faire des trucs après le chargement du fichier de settings local. Sinon on peut aussi faire des macros i.e. par exemple juste avant le execfile() du fichier PROJECT_SETTINGS_FILE on définit:

```
def debug():
   global DEBUG, LOGGING
   DEBUG = True
   for logger in LOGGING["loggers"]:
       logger["level"] = "DEBUG"
```

Comme cela on définit un comportement par défaut utilisable facilement mais on laisse aussi la possibilité de faire autrement.

- Les tracebacks sont absolument inutilisable dans les logs, quand sentry est utilisé ce n'est pas un souci mais dans l'absolu c'est gênant. Pour pallier à cela j'ai dévéloppé un handler dans le projet python-entrouvert nommé entrouvert.logging.handlers.SyslogHandler qui découpe les message de plus de 120 caractères en plusieurs pour que tout passe. Généralise-t-on son utilisation ou peut-on envisager une autre solution ?
- Logging par tenant ? A mon avis un objectif raisonnable ce serait de pouvoir ajouter un préfixe au début des messages pour pouvoir ensuite les filtrer avec grep ou les dispatcher avec rsyslog vers des fichiers individuels. On pourrait s'en sortir avec une classer Formatter qui ferait un get_connection() et aui ajouterait le nom du tenant au logrecord avant de formater le message.
- Il semblerait qu'il soit possible de ne pas utiliser le paramétrage par défaut de Django, qui est effectivement peu pratique, pour faire des logs, https://www.caktusgroup.com/blog/2015/01/27/Django-Logging-Configuration-logging_config-default-settings-logger/

Pour débugger:

- https://pypi.python.org/pypi/logging_tree

```
>>> from logging_tree import printout
>>> printout()
   ""
   Level DEBUG
   Propagate OFF
   Handler Stream <open file '<stderr>', mode 'w' at 0x7fd2285a2270>
     Level DEBUG
     Formatter fmt='[%(asctime)s] %(levelname)s %(name)s.%(funcName)s: %(message)s' datefmt='%Y-%m-%d %a %
H:%M:%S'
   Handler <authentic2.middleware.ThreadTrackingHandler object at 0x3430b10>
   |
   o<--"attribute_aggregator"
   |   Level NOTSET so inherits level DEBUG
   |
   o<--[authentic2]
   |   |
   |   o<--[authentic2.attribute_aggregator]
   |   |   |
   |   |   o<--"authentic2.attribute_aggregator.attributes"
   |   |   |   Level NOTSET so inherits level DEBUG
   |   |   |
   |   |   o<--"authentic2.attribute_aggregator.models"
   |   |   |   Level NOTSET so inherits level DEBUG
   |   |   |
   |   |   o<--"authentic2.attribute_aggregator.user_profile"
   |   |       Level NOTSET so inherits level DEBUG
   |   |
   |   o<--"authentic2.managers"
   |   |   Level NOTSET so inherits level DEBUG
   |   |
   |   o<--"authentic2.plugins"
   |   |   Level NOTSET so inherits level DEBUG
   |   |   Disabled
   |   |
   |   o<--[authentic2.saml]
   |       |
   |       o<--"authentic2.saml.admin"
   |           Level NOTSET so inherits level DEBUG
   |
   o<--[django]
   |   |
   |   o   "django.db"
   |   |   Level INFO
   |   |   Propagate OFF
   |   |   Handler Stream <open file '<stderr>', mode 'w' at 0x7fd2285a2270>
   |   |     Level DEBUG
   |   |     Formatter fmt='[%(asctime)s] %(levelname)s %(name)s.%(funcName)s: %(message)s' datefmt='%Y-%m
-%d %a %H:%M:%S'
   |   |   |
   |   |   o<--"django.db.backends"
   |   |       Level NOTSET so inherits level INFO
```

```
|   |       |
|   |       o<--"django.db.backends.schema"
|   |           Level NOTSET so inherits level INFO
|   |
|   o<--"django.request"
|       Level NOTSET so inherits level DEBUG
|       Disabled
|
o<--"django_select2"
|   Level NOTSET so inherits level DEBUG
|   |
|   o<--"django_select2.fields"
|   |   Level NOTSET so inherits level DEBUG
|   |
|   o<--"django_select2.util"
|   |   Level NOTSET so inherits level DEBUG
|   |
|   o<--"django_select2.widgets"
|       Level NOTSET so inherits level DEBUG
|
o<--[py]
|   |
|   o<--"py.warnings"
|       Level NOTSET so inherits level DEBUG
|       Handler <logging.NullHandler object at 0x20ed250>
|
o<--"requests"
    Level NOTSET so inherits level DEBUG
    Handler <logging.NullHandler object at 0x2f4a3d0>
    |
    o<--[requests.packages]
        |
        o<--"requests.packages.urllib3"
            Level NOTSET so inherits level DEBUG
            Handler <logging.NullHandler object at 0x2f4a350>
            |
            o<--"requests.packages.urllib3.connectionpool"
            |   Level NOTSET so inherits level DEBUG
            |
            o<--"requests.packages.urllib3.poolmanager"
            |   Level NOTSET so inherits level DEBUG
            |
            o<--[requests.packages.urllib3.util]
                |
                o<--"requests.packages.urllib3.util.retry"
                    Level NOTSET so inherits level DEBUG
```

testé sur authentic avec la configuration suggérée par les gens de Caktus group

- Hynek Schlawack - Beyond grep: Practical Logging and Metrics - PyCon 2015, https://www.youtube.com/watch?v=gqmAwK0wNyw (https://hynek.me/talks/beyond-grep/)

HowDoWeDoPostgreSQL

Tuning

Utiliser http://pgtune.leopard.in.ua/

Entrer RAM et nombre max de connections parallèles (= (nombre de processus uwsgi + gunicorn qui pointent vers la base) * 1.5):

Attention ces valeurs ne concernent qu'une machine postgresql seule! Si d'autre processus tournent sur la même machine il faut diminuer la RAM allouée à postgresql pour en garder pour les autres processus

Autres ressources:

- https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server
- http://thebuild.com/presentations/not-your-job-pgconf-us-2017.pdf : bonnes références notamment sur la configuration des logs pour trouver le bon work_mem
- https://www.citusdata.com/blog/2017/09/29/what-performance-can-you-expect-from-postgres/
- tuto rapide par nos copains d'Evolix : https://wiki.evolix.org/HowtoPostgreSQL
- super site sur les paramètres de conf postgresql : https://postgresqlco.nf/fr/doc/param/

Et sinon, appeler Dalibo.

Pour les devs

```
$ psql
> \l                # liste les bases
> \c passerelle     # se connecter à cette base
> \dn               # liste des schémas
> set search_path="passerelle_dev_publik_love";
> \dt               # liste des tables (ayant ce shcéma)
```

HowDoWeDoProvisioning

L'application agent hobo d'authentic2 écoute les signaux pre_{save/delete} des utilisateurs, rôles, relations de parenté entre rôles et relation d'enrôlement entre utilisateurs et rôles. En fin de requête un thread est lancé qui génère des messages de type "notify" poussé en AMQP (via RabbitMQ) pour les autres services.

Les agents reçoivent et agissent en conséquence.

La clé unique de communication, c'est l'uuid pour les rôles et les utilisateurs, néanmoins pour des raisons de reprise si l'uuid n'est pas trouvé on essaie de retrouver via le slug ou le nom dans le cas des rôles.

Dans les applications utilisant django-mellon, le mapping entre utilisateur NameID est stocké via le modèle UserSAMLIdentifier, le username est aussi généré en tronquant le NameID à 31 caractères; et c'est mappé sur un champ uuid d'un nouveau modèle Role (class Role(Group)) pour les rôles.

Par ailleurs, lors du SSO, l'assertion contient un attribut role-slug qui contient la liste des uuid de rôles.

HowDoWeDoPublikDomainNameChange (pas sur le SaaS)

DNS

S'assurer que les noms de domaine existent. Sur un hébérgement on-premise, si besoin, taper les noms dans /etc/hosts.

Certificats

Obtenir le certificat pour le nouveau domaine. Placer dans /etc/ssl/{private,certs} ou /etc/nginx/ssl/.

NGINX

Créer le fichier de conf déclarant le nouveau certificat et sa clé. Ex: /etc/nginx/includes/wildcard.demarches.grenoblealpesmetropole.fr.conf

```
ssl_certificate   /etc/nginx/ssl/wildcard.demarches.grenoblealpesmetropole.fr.crt;
ssl_certificate_key   /etc/nginx/ssl/wildcard.demarches.grenoblealpesmetropole.fr.key;
```

Page de maintenance

- prévoir la modification de la conf nginx pour déclarer les noms de domaine supplémentaires:
  /etc/nginx/conf.d/server-names-hash-bucket-size.conf:
  server_names_hash_bucket_size 128;
- rajouter le nouveau nom de domaine dans les directions de port 80 => 443

```
   server {
          listen   80;
          server_name  statistiques.demarches.lametro.fr statistiques.demarches.grenoblealpesmetropole.fr;

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

          return 302 https://$host$request_uri;
   }
```

Configurer le mode maintenance de Nginx:

Créer un fichier de configuration du mode maintenance: /etc/nginx/includes/maintenance.conf

```
    if (-f $document_root/maintenance.html) {
        set $maintenance 1;
    }

    if ($maintenance = 1) {
         return 503;
    }

    error_page 503 @maintenance;

    location @maintenance {
         rewrite ^(.*)$ /maintenance.html break;
    }
```

Créer le fichier /usr/share/nginx/html/maintenance.html et y poser le message de maintenance en HTML.

Déclarer un vhost pour l'ancien nom de domaine, avec la conf SSL, et de la page de maintenance. Ex:

```
server {
     listen 443 ssl;
     server_name accueil.demarches.lametro.fr;

     include includes/ssl.conf;
     include includes/wildcard.demarches.lametro.fr.conf;
```

```
        include includes/maintenance.conf;
    }
```

Test de la config:

```
sudo nginx -t
```

et reload si tout va bien.

Services

- stop services:

  ```
  sudo systemctl stop authentic2-multitenant bijoe chrono combo fargo hobo passerelle wcs welco
  ```

- stop cron jobs:

  ```
  sudo vi /etc/cron.d/wcs
  ```

Tenants

Rénommer les répértoires des tenants:

```
  sudo -u authentic-multitenant mv /var/lib/authentic2-multitenant/tenants/connexion.demarches.lametro.fr{,.in
valid.migration}
  sudo -u bijoe mv /var/lib/bijoe/tenants/statistiques.demarches.lametro.fr{,.invalid.migration}
  sudo -u chrono mv /var/lib/chrono/tenants/agendas.demarches.lametro.fr{,.invalid.migration}
  sudo -u combo mv /var/lib/combo/tenants/portail-agent.demarches.lametro.fr{,.invalid.migration}
  sudo -u combo mv /var/lib/combo/tenants/demarches.lametro.fr{,.invalid.migration}
  sudo -u hobo mv /var/lib/hobo/tenants/hobo.demarches.lametro.fr{,.invalid.migration}
  sudo -u fargo mv /var/lib/fargo/tenants/porte-documents.demarches.lametro.fr{,.invalid.migration}
  sudo -u passerelle mv /var/lib/passerelle/tenants/passerelle.demarches.lametro.fr{,.invalid.migration}
  sudo -u wcs mv /var/lib/wcs/services.demarches.lametro.fr{,.invalid.migration}
  sudo -u welco mv /var/lib/welco/tenants/accueil.demarches.lametro.fr{,.invalid.migration}
```

Schémas des bases des données

Exporter les schemas pour toutes les briques, sauf wcs, en remplaçant l'ancien nom de domaine par le nouveau:

- hobo:

  ```
  sudo -u hobo pg_dump -n hobo_demarches_lametro_fr | sed 's/demarches_lametro_fr/demarches_grenoblealpesmet
  ropole_fr/g' | sed 's/demarches.lametro.fr/demarches.grenoblealpesmetropole.fr/g'  > hobo.sql
  ```

- combo:

  ```
  sudo -u combo pg_dump -n demarches_lametro_fr -n portail_agent_demarches_lametro_fr | sed 's/demarches_lam
  etro_fr/demarches_grenoblealpesmetropole_fr/g' | sed 's/demarches.lametro.fr/demarches.grenoblealpesmetrop
  ole.fr/g' > combo.sql
  ```

- passerelle:

  ```
  sudo -u passerelle pg_dump -n passerelle_demarches_lametro_fr | sed 's/demarches_lametro_fr/demarches_gren
  ```

```
oblealpesmetropole_fr/g' | sed 's/demarches.lametro.fr/demarches.grenoblealpesmetropole.fr/g'  > passerell
e.sql
```

- welco:

```
sudo -u welco pg_dump -n accueil_demarches_lametro_fr | sed 's/demarches_lametro_fr/demarches_grenoblealpe
smetropole_fr/g' | sed 's/demarches.lametro.fr/demarches.grenoblealpesmetropole.fr/g' > welco.sql
```

- authentic:

```
sudo -u authentic-multitenant pg_dump -n connexion_demarches_lametro_fr authentic2_multitenant | sed 's/de
marches_lametro_fr/demarches_grenoblealpesmetropole_fr/g' | sed 's/demarches.lametro.fr/demarches.grenoble
alpesmetropole.fr/g'  > authentic.sql
```

- fargo:

```
sudo -u fargo pg_dump -n porte_documents_demarches_lametro_fr | sed 's/demarches_lametro_fr/demarches_gren
oblealpesmetropole_fr/g' | sed 's/demarches.lametro.fr/demarches.grenoblealpesmetropole.fr/g' > fargo.sql
```

- chrono:

```
sudo -u chrono pg_dump -n agendas_demarches_lametro_fr | sed 's/demarches_lametro_fr/demarches_grenoblealp
esmetropole_fr/g' | sed 's/demarches.lametro.fr/demarches.grenoblealpesmetropole.fr/g' > chrono.sql
```

- bijoe:

```
sudo -u bijoe pg_dump -n services_demarches_lametro_fr -n statistiques_demarches_lametro_fr | sed 's/demar
ches_lametro_fr/demarches_grenoblealpesmetropole_fr/g' | sed 's/demarches.lametro.fr/demarches.grenoblealp
esmetropole.fr/g'  > bijoe.sql
```

cela permet d'automatiquement renommer les issues dans mellon.UserSAMLIdentifier, les métadonnées des SP dans la base d'Authentic.

Pour chaque service

Authentic

- modifier le nom de domaine dans
  - hobo.json
  - settings.json (s'il existe) /!\ attention à ne pas modifier le nom de domaine dans la conf LDAP s'il est utilisé dans le realm

Combo, Welco, Passerelle, Chrono, Fargo, Bijoe

- modifier le nom de domaine dans
  - idp-metadata-1.xml
  - hobo.json

Hobo

- modifier le nom de domaine dans
  - idp-metadata-1.xml
  - base_url

Bijoe

- modifier le nom de domaine dans
  - wcs-olap.ini
  - schemas/services_demarches_lametro_fr.model
  - sudo -u bijoe mv schemas/services_demarches_{lametro,grenoblealpesmetropole}_fr.model

WCS

Renommer les fichiers avec les métadonnées d'authentic

```
mv idp-https-connexion.demarches.lametro.fr-idp-saml2-metadata-metadata.xml idp-https-connexion.demarches.grenoblealpesmetropole.fr-idp-saml2-metadata-metadata.xml
```

et y remplacer l'ancien nom de domaine par le nouveau. Ex: lametro par grenoblealpesmetropole

- dans saml2-metadata.xml remplacer l'ancien nom de domaine par le nouveau: Ex: lametro par grenoblealpesmetropole
- dans site-options.cfg remplacer l'ancien nom de domaine par le nouveau: Ex: lametro par grenoblealpesmetropole
  - [options]
    - modifier: theme_skeleton_url
  - [variables]
    - modifier: idp_url
    - modifier: portal_user_url
    - modifier: portal_agent_url
    - etc...

Mettre à jour le fichier config.pck du tenant (en faisant une sauvegarde au préalable):

```
import pickle
data = pickle.load(file('config.pck'))

data['idp']['https-connexion.demarches.grenoblealpesmetropole.fr-idp-saml2-metadata'] = data['idp']['https-connexion.demarches.lametro.fr-idp-saml2-metadata']
data['idp']['https-connexion.demarches.grenoblealpesmetropole.fr-idp-saml2-metadata']['metadata'] = data['idp']['https-connexion.demarches.grenoblealpesmetropole.fr-idp-saml2-metadata']['metadata'].replace('lametro', 'grenoblealpesmetropole')
data['idp']['https-connexion.demarches.grenoblealpesmetropole.fr-idp-saml2-metadata']['metadata_url'] = data['idp']['https-connexion.demarches.grenoblealpesmetropole.fr-idp-saml2-metadata']['metadata_url'].replace('lametro', 'grenoblealpesmetropole')
data['idp'].pop('https-connexion.demarches.lametro.fr-idp-saml2-metadata')

data['saml_identities']['registration-url'] = data['saml_identities']['registration-url'].replace('lametro', 'grenoblealpesmetropole')

data['sp']['saml2_base_url'] = data['sp']['saml2_base_url'].replace('lametro', 'grenoblealpesmetropole')
data['sp']['saml2_providerid'] = data['sp']['saml2_providerid'].replace('lametro', 'grenoblealpesmetropole')

data['misc']['frontoffice-url'] = data['misc']['frontoffice-url'].replace('lametro', 'grenoblealpesmetropole')
data['misc']['homepage-redirect-url'] = data['misc']['homepage-redirect-url'].replace('lametro', 'grenoblealpesmetropole')

# optionnel
data['postgresql']['database'] = data['postgresql']['database'].replace('lametro', 'grenoblealpesmetropole')

with open('new_config.pck', 'w') as conf:
    pickle.dump(data, conf)
```

Vérifier les dans /formsdefs/, /workflows/, /mail-templates/ s'il y a des références en dur vers le nom de domaine et le remplacer.

Création des nouveaux schémas

Importer les dumps créés plus haut:

```
sudo -u hobo psql < hobo.sql
sudo -u combo psql < combo.sql
sudo -u passerelle psql < passerelle.sql
sudo -u welco psql < welco.sql
sudo -u authentic-multitenant psql authentic2_multitenant < authentic.sql
sudo -u fargo psql < fargo.sql
sudo -u chrono psql < chrono.sql
sudo -u bijoe psql < bijoe.sql
```

Renommer les répértoires des tenants

```
sudo -u authentic-multitenant mv /var/lib/authentic2-multitenant/tenants/connexion.demarches.lametro.fr.invalid.migration /var/lib/authentic2-multitenant/tenants/connexion.demarches.grenoblealpesmetropole.fr
sudo -u bijoe mv /var/lib/bijoe/tenants/statistiques.demarches.lametro.fr.invalid.migration /var/lib/bijoe/tenants/statistiques.demarches.grenoblealpesmetropole.fr
sudo -u chrono mv /var/lib/chrono/tenants/agendas.demarches.lametro.fr.invalid.migration /var/lib/chrono/tenants/agendas.demarches.grenoblealpesmetropole.fr
sudo -u combo mv /var/lib/combo/tenants/portail-agent.demarches.lametro.fr.invalid.migration /var/lib/combo/tenants/portail-agent.demarches.grenoblealpesmetropole.fr
sudo -u combo mv /var/lib/combo/tenants/demarches.lametro.fr.invalid.migration /var/lib/combo/tenants/demarches.grenoblealpesmetropole.fr
sudo -u hobo mv /var/lib/hobo/tenants/hobo.demarches.lametro.fr.invalid.migration /var/lib/hobo/tenants/hobo.demarches.grenoblealpesmetropole.fr
sudo -u fargo mv /var/lib/fargo/tenants/porte-documents.demarches.lametro.fr.invalid.migration /var/lib/fargo/tenants/porte-documents.demarches.grenoblealpesmetropole.fr
sudo -u passerelle mv /var/lib/passerelle/tenants/passerelle.demarches.lametro.fr.invalid.migration /var/lib/passerelle/tenants/passerelle.demarches.grenoblealpesmetropole.fr
sudo -u wcs mv /var/lib/wcs/services.demarches.lametro.fr.invalid.migration /var/lib/wcs/services.demarches.grenoblealpesmetropole.fr
sudo -u welco mv /var/lib/welco/tenants/accueil.demarches.lametro.fr.invalid.migration /var/lib/welco/tenants/accueil.demarches.grenoblealpesmetropole.fr
```

Démarrer les services et les crons:

```
sudo systemctl start authentic2-multitenant bijoe chrono combo fargo hobo passerelle wcs welco
```

```
sudo vi /etc/cron.d/wcs
```

Retrait de la page de maintenance et redirection vers le nouveau domaine:

```
  server {
       listen 443 ssl;
       server_name accueil.demarches.lametro.fr;

       include includes/ssl.conf;
       include includes/wildcard.demarches.lametro.fr.conf;

       # include includes/maintenance.conf;
       return 302 https://accueil.demarches.lametro.fr$request_uri;
  }
```

Test de la config:

```
sudo nginx -t
```

et reload si tout va bien.

Warnings

FranceConnect

Il suffit de déclarer les nouvelles URLs de retour (callback et redirection de déconnexion) en plus des anciennes, puis de retirer les anciennes un fois la migration faite. Il n'y a pas de validation faites sur ces URLs dans le back-office de FranceConnect donc pas de blocage possible.

wcs

Les demandes peuvent contenir des URLs avec l'ancien nom de domaine (typiquement les URLs vers les résas dans Chrono). Les appels webservice vers ces URLs vont échouer car ne seront pas signés.
Il faut identifier toutes les demandes ayant des URLs avec l'ancien nom de domaine dans data et workflow_data et mettre à jour.

Changement du nom de domaine de base d’une instance de Publik sur le SaaS

DNS / Certificats

Puppet etc.

- TODO lien vers une page qui expliquerait où/comment, vite fait prod : modules/haproxy/files/prod.saas.entrouvert.org/bundles/
- TODO info dans le cas de let's encrypt

haproxy

Puppet etc. pareil qu'un déploiement, i.e. modif publik.map, ex: modules/haproxy/files/prod.saas.entrouvert.org/publik.map

Pointage des noms des nouveaux tenants vers les schémas db actuel

/etc/$module/settings.d/schema_names.py

```
TENANT_MAPPING = {
  "nom.du.nouveau.domaine": "nom_de_l_ancien_schema",
  ...
}
```

Interlude : début downtime

Puppet pour modification haproxy pour afficher une page de maintenance sur les anciens noms de domaine,

```
demarches.maville.fr               maintenance
agents.demarches.maville.fr        maintenance
connexion.demarches.maville.fr     maintenance
formulaires.demarches.maville.fr   maintenance
passerelle.demarches.maville.fr    maintenance
hobo.demarches.maville.fr          maintenance
porte-doc.demarches.maville.fr     maintenance
agendas.demarches.maville.fr       maintenance
statistiques.demarches.maville.fr  maintenance
```

- TODO: implémenter ça. (#54379).

Mise en place des tenants avec les nouveaux noms

Dans /var/lib/$module/tenants/ de chacun, renommer le répertoire du tenant vers le nouveau nom.

Adapter les configurations qui contiendraient l’ancien nom

Sur les services dans la db de hobo et la db d'authentic et les hobo.json et idp-metadata-1.xml et d'autres encore peut-être.

- TODO: faire le tour
- TODO: réduire la liste parce que assurer que l'étape suivante fasse un max
  - par exemple #54380 pour w.c.s.

Exécuter un cook recipe.json des nouveaux noms

Ça mettra carré des infos dans le site-options.cfg de w.c.s. et sans doute d'autres choses.

Adapter les données qui contiendraient l’ancien nom

Genre des URL absolues dans des cellules texte, dans des demandes, etc.

Interlude : fin downtime

Puppet pour modification haproxy pour assurer des redirections des anciens noms de domaine vers les nouveaux. (en gardant les chemins, histoire que les anciennes URL continuent à fonctionner).

Postface

Dans HowDoWeDoPublikDomainNameChange il est écrit pour FranceConnect :

S'assurer que côté FranceConnect il y a juste changement d'URLs d'authentic et non la création d'un nouveau compte avec des nouveaux client_id et client_secret. Cela permet (en théorie) d'avoir les mêmes subs pour les comptes déjà fédérés.

HowDoWeDoPythonPackaging

Pousser sur pypi

Mettre dans ~/.pypirc

```
[distutils]
username: entrouvert
password: voir sur https://dev.entrouvert.org/projects/sysadmin/wiki/Mots_de_passe#Pypi
```

Installer twine :

```
apt install twine
```

Ensuite il faut construire la distribution des sources et binaire, pour un paquet mixte Python2/3 :

```
python setup.py sdist bdist_wheel --universal
```

Pour un paquet pure python2 ou 3 on enlèvera le --universal.

Ensuite il faut uploader les archives :

```
twine upload dist/*
```

HowDoWeDoSubmitFormsToWebservices

Dates

Les valeurs _raw des champs de type **date** dans wcs sont automatiquement sérialisées lors de l'envoi au webservice.
De ce fait, le paramètre doit être configuré de la sorte:

Données à envoyer dans le corps de la requête	
event_date	=form_var_event_date_raw

HowDoWeDoTests

- pytest, pytest-django et webtest: OK pour tout le monde
- tests/ à la racine, plutôt que distribués dans les différents répertoires des applications django (ça semble être une tendance). : NOK en django les tests vont dans les apps quand il y en a, OK pour les projets
- en Django prévoir un fichier settings de test pour les applications django (pour les projets le fichier settings normal suffit), ajouter une cible "test" au fichier setup.py qui appelle "django-admin test" ou pytest en installant le fichier de config
- utiliser tox pour tester dans différents environnements (django 1.6, 1.7, etc.. python 2.6, 2.7, 3.2, etc.., avec différentes lib python si il y a des options ou si le paquets est censé fonctionner avec ou sans une certaine dépendance) et plus généralement pour sa capacité à gérer tout seul la création d'un virtualenv adapté ce qui permet de détecter les soucis de packaging (dépendance manquante ou dont la dernière version n'est plus compatible)

Ressources

- mock/requests: http://engineroom.trackmaven.com/blog/real-life-mocking/
- pytest + webtest + Django : http://mathieu.agopian.info/presentations/2013_09_djangocong/

Tests manuels

- Pour tester un callback HTTP ou simplement un appel: http://requestbin.com/ (API carte-bancaire par exemple)
- Pour tester la réaction à des réponses HTTP bizarres: http://httpbin.org/
- Pour tester l'envoi de mail sans pourrir sa boite: https://one-time.email/ http://getairmail.com/ (pas pour des trucs dangereux hein ! genre récupérer son compte sur un site en prod :))

Pense bête py.test&Django

Une fois qu'on a tox de configuré pour lancer les tests py.test on peut faire pas mal de choses amusantes, je donne mes exemples dans le contexte du projet authentic2.

- lancer les tests dans un environnement donné (parce qu'on a pas besoin de tout tester là maintenant, ça c'est pour jenkins)

```
$ tox -e dj18-authentic-sqlite
```

- lancer les tests en conservant la base de donnée (ça permet de passer l'étape laborieuse des migrations)

```
$ tox -e dj18-authentic-sqlite -- tests --reuse-db
```

- s'arrêter avec pdb dans le dernier test qui a planté (--lf pour lastfail et --pdb pour ouvrir pdb sur une exception)

```
$ tox -e dj18-authentic-sqlite -- tests --lf --reuse-db --pdb
```

- ne lancer qu'un test en particulier (on peut passer à l'option -k n'importe quelle sous chaîne pertinente du nom du ou des fonctions de test)

```
$ tox -e dj18-authentic-sqlite -- tests -k test_moncul --reuse-db --pdb
```

- recréer la base de test (parce qu'on a ajouté une migration)

```
$ tox -e dj18-authentic-sqlite -- tests --reuse-db --create-db
```

Utiliser un debugger alternatif

Par défaut, utiliser l'option --pdb ou taper import pdb; pdb.set_trace() dans le code ouvrira l'interpréteur pdb de base.
Il est possible d'utiliser un autre interpréteur, par exemple IPython.

L'environnement installé par tox n'inclut pas IPython. Il faut donc injecter globalement la dépendance via un plugin : pip install tox-ipdb-plugin.

Ensuite, il faut dire à pytest d'utiliser cet interpréteur pour pdb. Là encore un peu de gymnastique pour que la configuration soit globale, il faut ajouter deux variables d'environnement dans son .bashrc :

```
export PYTEST_ADDOPTS='--pdbcls=IPython.terminal.debugger:TerminalPdb'
```

```
export TOX_TESTENV_PASSENV='PYTEST_ADDOPTS'
```

Voilà, vive l'autocomplétion.

HowDoWeDoThemes

(work in progress)

Le dévéloppement d'une intégration graphique exige un environnement de développement local, cf https://doc-publik.entrouvert.com/dev/installation-developpeur/

publik-base-theme

Le module publik-base-theme malgré son nom est le dépôt de (presque) toutes nos intégrations graphiques.

La liste est établie (lors de l'appel à make) en parcourant les répertoires sous static/, en tirant les informations de fichiers config.json, ex static/publik/config.json :

```
{
  "label": "Publik",
  "variables": {
    "theme_color": "#E80E89"
  }
}
```

- Un identifiant est créé automatiquement en prenant le nom du répertoire.
- Le libellé sert à la sélection du thème dans l'écran dédié dans Hobo.
- La variable theme_color reprend la couleur dominante du thème, elle est affichée dans l'écran dans Hobo mais également posée dans des balises <meta> pour colorer l'UI (chrome) de certains navigateurs web (mobile).
- Il peut aussi y avoir une variable favicon, contenant le chemin vers la favicon (ex: fondettes/favicon.png pour /static/fondettes/favicon.png).
- Une variable logo_link_url pour que le lien primaire du titre/logo de la page ne pointe pas vers le portail citoyen mais vers ailleurs.

Style CSS

Dans publik-base-theme/static, il y a donc un répertoire par intégration. Celui-ci doit contenir un fichier style.css. On utilise le préprocesseur sass pour le créer à partir d'un fichier style.scss, ce travail est assuré via des règles dans le Makefile, la cible pour la création des fichiers style.css s'appelle "css" → make css exécuté depuis le répertoire publik-base-theme créera les fichiers style.css des différentes intégrations. Il est bien sûr également possible de viser un seul fichier, ex: make static/publik/style.css.

Le fichier style.scss sert à importer différents fichiers, généralement :

```
@import 'vars';
@import '../includes/publik';
@import 'custom';
```

La définition des styles de Publik vient de ../includes/publik, les deux autres fichiers servent à définir les spécificités locales.

Le fichier _vars.scss peut définir une série de variables (liste complète publiée sur https://doc.entrouvert.org/publik-base-theme/dev/misc-scss.html) qui permettent de paramétrer le rendu général du site, généralement en y appliquant les valeurs tirées d'une maquette ou du site servant de modèle.

```
@charset "UTF-8";

$primary-color: #1DA1AE;

$font-color: #565656;
$font-size: 13px;
$font-family: sans-serif;
$nav-background: $primary-color;
$nav-color: white;
$nav-active-color: darken($primary-color, 20%);
$border-radius: 3px;
$button-background: $primary-color;
$title-background: $primary-color;
$title-color: white;
$footer-background: transparent;
$footer-color: $font-color;
```

Pour aller au-delà de ce que ce paramétrage permet; le fichier _custom.scss peut accueillir des règles CSS supplémentaires :

```
#header {
```

```
    background: url(images/bandeau.jpg) no-repeat scroll left bottom;
    height: 180px;
}

body {
    background: url(images/bg_header.png) repeat-x scroll 0 2px;
}
[...]
```

Si on se trouve à copier/coller ou répéter des trucs de _custom.scss en _custom.scss, on doit sans doute réfléchir et voir pour intégrer ça dans les fichiers scss partagés.

Documentation publique des classes css

- à lire ici au préalable pour ne pas doublonner de l'existant : https://doc-publik.entrouvert.com/admin-fonctionnel/modifier-le-contenu-des-portails/

Templates

Les templates Django sont dans le répertoire templates/; ils surchargent une partie des templates natifs des différents modules, pour diverses raisons (spécificités à Publik, paresse, etc.).

À éviter, il y a la possibilité pour un thème, de redéfinir de manière spécifique un template donné, en le plaçant sous templates/variants/$theme/, par exemple templates/variants/nancy-2017/combo/wcs/forms_of_category.html.

Images

Elles sont normalement rangées dans le répertoire img/ (mais ce n'est pas encore systématique).

Pour des petites images, il peut être intéressant d'optimiser le chargement en incluant directement l'image dans le fichier CSS, en utilisant une URI data. Pour assurer ce travail il y a une cible data_uris dans le Makefile, qui appelle le script make_data_uris.py sur un répertoire donné, ex :

```
python make_data_uris.py static/grandlyon-gnm/
```

Cela crée un fichier _data_uris.scss qui peut être inclus dans le thème et les images peuvent alors être référencées ainsi : url($data_uri_cadre_de_vie) (data_uri_ + le nom de fichier sans l'extension).

Icônes du tableau de bord

Inclure ../includes/dashboard dans les scss et ajouter le nécessaire au Makefile pour générer les images dans les couleurs désirées, ex :

```
cd src/ && python render-imgs-dashboard.py ../static/chateauroux/img/ --normal 333333 --selected 0779B7 --titl
e FFFFFF --title-width 80
```

Icônes des catégories

Inclure ../includes/categories dans les scss et ajouter au Makefile.

```
cd src/ && python render-imgs-categories.py ../static/orleans/img/ --primary f05923 --secondary 34697D
```

Polices de caractère

On évite la récupération de polices depuis les CDN de Google ou autre, on incorpore directement dans publik-base-theme les polices (pour lesquelles les licences autorisent cela, bien sûr). On se trouve ainsi dans static/includes/ avec une série de fichiers _font-*.scss, ex: _font-montserrat.scss.

Pour donner accès à une police, on ajoutera donc un import de la forme :

```
@import '../includes/font-montserrat';
```

On inclut uniquement les versions woff2 et woff car c'est désormais pris en charge "partout" (>IE8).

Détails de la mécanique bas niveau

(Cette section n'est pas nécessaire pour la réalisation de thèmes.)

On définit un thème pour Combo ; d'abord de manière classique, par exemple un "common.html" qui va de <html> à </html> et puis les différents modèles de page, "page_template.html" et "page_template_sidebar.html" etc. qui {% extends "common.html" %}.

On précise les URL vers des ressources (css, js, images…) de manière absolue, il y a pour ça un {{site_base}} qui est utile; ex :

```
<script src="{{site_base}}{% xstatic 'jquery' 'jquery.min.js' %}"></script>
<script src="{{site_base}}{% static 'js/combo.public.js' %}"></script>
```

Ensuite, les fichiers base.html des applications peuvent être ajoutés (idéalement les applications cherchent toutes d'abord un /$app/base.html, donc les templates ne se mangent pas l'espace du /base.html). Ces fichiers démarrent sur un modèle simple, d'une ligne, {% extends theme_base %}. Cela créera à la volée un template avec un squelette généré sur base de celui servi par combo.

Pour ce faire l'application (en fait un context processor d'hobo) se base sur un paramètre THEME_SKELETON_URL, qui pointe vers le combo, ex :

```
THEME_SKELETON_URL = 'http://combo-test-theme.127.0.0.1.xip.io:8000/__skeleton__/'
```

Le context processor ajoute l'url demandée (paramètre source) et avec ça Combo trouve la page correspondant à l'application (en matchant le mieux possible l'url donnée, ce qui permet par exemple d'avoir une page pointant vers http://wcs.example.net/etat-civil/ et une autre pointant vers http://wcs/sport/, et y avoir des différences de contenu). Combo fait le rendu de cette page en remplaçant les zones laissées vides (sans cellules) par des appels de bloc django (ex : {% block content }{ endblock %}), ceux-ci seront alors interprété quand le context processor créera un objet Template() à partir du contenu de la page ainsi récupérée.

C'est possible d'ajouter des zones qui donneront des blocs sans pour autant donner des zones présentées à la composition des pages dans combo, il faut utiliser le templatetag skeleton_extra_placeholder, ex : {% skeleton_extra_placeholder 'extra-head' } ou { skeleton_extra_placeholder 'extra-body-args' %}; c'est ainsi possible de créer une structure correspondant à celle attendue par l'application.

Elle pourrait être composée directement au niveau de combo mais ça peut alourdir mochement le code, par exemple si on a authentic qui attend body-args et une autre qui atend autre-body-args, et ainsi de suite :

```
<body {% block body-args %}{% endblock %} {% block autre-body-args %}{% endblock %} ...>
```

Pour éviter ça, on peut mettre un seul bloc, et dans le $app/base.html, prolonger le extends par un mapping des blocs :

```
{% extends theme_base %}
{% block extra-body-args %}
  {% block bodyargs %}
  {% endblock %}
{% endblock %}
```

Définition intégration "light"

- pas de template
- un _custom.scss de max 100 lignes

Mise en recettes et release du paquet deb publik-base-theme

Ça se passe dans de dépot publik-base-theme

1. un fois ses commits validés, on peut pousser sur le master
2. Jenkins va faire construire à partir du master à intervalles réguliers, et créer un paquet deb dans le dépôt 'unstable' d'EO
   - vérifiez que tout se fini bien ici : https://jenkins.entrouvert.org/job/publik-base-theme-deb/
3. si les autres développeurs sont d'accords, on peut taguer une nouvelle version du dépôt Git : git tag -a v2.22
4. Jenkins va aussi construire à partir du tag et créer un paquet de dans le dépôt 'testing' d'EO
   - vérifiez aussi que tout se passe bien
5. on peut alors mettre à jour les recettes sur **toutes les machines**, on peut exécuter apt-get update && apt-get upgrade publik-base-theme
   - liste de toutes les machines de recette https://dev.entrouvert.org/projects/sysadmin/wiki/Mises_%C3%A0_jour_Publik#Recettes
   - on peut utiliser l'outil eotasks pour exécuter les commandes en parallèle sur les serveurs: [[sysadmin:Exécution_de_taches_sur_les_serveurs]]

HowDoWeDoTickets

Pour poser du code, genre une trace, on l'insère entre balises "pre".

Pour citer quoi que ce soit, genre un passage d'un autre ticket, on le préfixe de >.

HowDoWeDoUpgrades

Dépôts

Nos paquets sont distribués via des dépôts signés : https://deb.entrouvert.org/

Vous pouvez accepter notre clé GPG avec l'empreinte 27FF12846574F8A0EFFF7A84DF69CC342CCFEC25 en exécutant la commande suivante :

```
wget -O - https://deb.entrouvert.org/entrouvert.gpg | sudo apt-key add -
```

Configuration de apt

Afin de configurer apt pour récupérer les versions adaptées des logiciels issus des dépôts officiels, nous installons le paquet entrouvert-repository :

```
sudo apt install entrouvert-repository
sudo apt update
```

Mises-à-jour standards

Tous les seconds et quatrièmes jeudi nous livrons des mises-à-jours de tous les composants logiciels de Publik.

Pour les installer il est nécessaire d'executer:

```
sudo apt update
sudo apt full-upgrade
```

Mises-à-jour majeures : passage de Debian 8 ("Jessie") à Debian 9 ("Stretch")

1. Nettoyer les fichiers de préférences apt qui concernent Debian Jessie.

2. Adapter manuellement la configuration des dépôts :

```
sed 's/jessie/stretch/' -i /etc/apt/sources.list
sed 's/jessie/stretch/' -i /etc/apt/sources.list.d/entrouvert.list
apt update
```

S'assurer que le depot stretch-backports est bien présent dans /etc/apt/sources.list. **Il est obligatoire pour avoir la version 1.11 de Django.**

Supprimer d'éventuelles références additionnelles à nos dépôts.

3. Configurer apt:

```
apt install entrouvert-repository
# pour un serveur de recette, installer également entrouvert-repository-testing
apt update
```

4. Installer les mises-à-jours :

Attention: lors de la mise à jour du serveur openssh ne pas écraser le fichier de conf avec celui du paquet, pour ne pas changer le port d'écoute. (de manière générale, ne pas écraser les configs qui auraient été modifiées)

```
apt full-upgrade
```

5. Si le serveur tourne w.c.s. vérifier l'installation de libreoffice :

```
apt install libreoffice
```

6. Migrer de version PostgreSQL (vers 9.6)

En utilisant pg_upgradecluster, cf la documentation du paquet postgresql pour les détails d'utilisation.

7. Relancer les services :

```
systemctl restart authentic2-multitenant bijoe chrono combo corbo fargo hobo passerelle wcs
```

Mises-à-jour majeures : passage de Debian 9 ("Stretch") à Debian 10 ("Buster")

Préliminaire : la machine doit être parfaitement à jour. Lancer tout d'abord une mise à jour classique « apt update && apt full-upgrade »

1. Nettoyer les fichiers de préférences apt qui concernent Debian Buster et inférieurs.

2. Adapter manuellement la configuration des dépôts :

```
# sed 's/stretch/buster/' -i /etc/apt/sources.list
```

S'assurer que le depot buster-backports est bien présent dans /etc/apt/sources.list.d/.

3. Configurer apt:

```
# apt install entrouvert-archive-keyring entrouvert-repository entrouvert-repository-hotfix
## pour un serveur de recette : apt install entrouvert-repository-testing
# apt update
```

4. Commencer par mettre à jour postgresql et réindexer les bases, ensuite installer les autres mises-à-jours :

```
# apt update
# apt install postgresql
# sudo -u postgres reindexdb --all
# apt full-upgrade
```

5. Migrer le cluster PostgreSQL vers la version 11

En utilisant pg_upgradecluster, cf la documentation du paquet postgresql pour les détails d'utilisation.

HowDoWeKi (ah ah)

Quand on fait une modification on en écrit un résumé/description dans le champ "commentaire", comme ça c'est mentionné dans les emails de notification et dans l'historique.