RFC 8493 : The BagIt File Packaging Format (V1.0)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 20 décembre 2018

Date de publication du RFC : Octobre 2018

https://www.bortzmeyer.org/8493.html

Le format BagIt, très utilisé dans le monde des bibliothèques (monde d'où sont issus les auteurs de ce RFC), décrit une série de conventions pour un ensemble de fichiers décrivant un contenu numérique quelconque. En fait, BagIt n'est pas vraiment un format (on ne peut pas le comparer à tar ou à zip), il définit juste les fichiers qui doivent être présents dans l'archive.

Une archive BagIt est appelée un **sac** ("*bag*"). Elle est composée des fichiers de contenu, qui sont d'un format quelconque, et des fichiers de métadonnées, qui décrivent le contenu (ces fichiers de métadonnées se nomment "*tags*"). BagTit met l'accent sur le contrôle de l'intégrité des données (les "*tags*" contiennent un condensat cryptographique des données) et sur la facilité d'accès à un fichier donné (les fichiers de données ne sont pas sérialisés dans un seul grand fichier, comme avec tar ou zip, ils restent sous la forme d'une arborescence).

La section 2 du RFC décrit la structure d'un sac :
— Des fichiers de métadonnées (les "*tags*") dont deux sont obligatoires, `bagit.txt` qui indique le numéro de version BagIt, et `manifest-HASHALGO.txt` qui contient les condensats.
— Un répertoire `data/` sous lequel se trouvent les fichiers de données.
— Si on veut, des répetoires contenant des fichiers de métadonnées optionnels.

Voici un exemple d'un sac, contenant deux fichiers de données, `Makefile` et `bortzmeyer-ripe-atlas-lapaz.tex` :

```
% find /tmp/RIPE-Atlas-Bolivia
/tmp/RIPE-Atlas-Bolivia
/tmp/RIPE-Atlas-Bolivia/manifest-sha512.txt
/tmp/RIPE-Atlas-Bolivia/data
/tmp/RIPE-Atlas-Bolivia/data/Makefile
/tmp/RIPE-Atlas-Bolivia/data/bortzmeyer-ripe-atlas-lapaz.tex
/tmp/RIPE-Atlas-Bolivia/manifest-sha256.txt
/tmp/RIPE-Atlas-Bolivia/tagmanifest-sha256.txt
/tmp/RIPE-Atlas-Bolivia/bag-info.txt
/tmp/RIPE-Atlas-Bolivia/tagmanifest-sha512.txt
```

```
/tmp/RIPE-Atlas-Bolivia/bagit.txt

% cat /tmp/RIPE-Atlas-Bolivia/bagit.txt
BagIt-Version: 0.97
Tag-File-Character-Encoding: UTF-8

% cat /tmp/RIPE-Atlas-Bolivia/bag-info.txt
Bag-Software-Agent: bagit.py v1.7.0 <https://github.com/LibraryOfCongress/bagit-python>
Bagging-Date: 2018-12-20
Contact-Email: stephane+atlas@bortzmeyer.org
Contact-Name: Stéphane Bortzmeyer
Payload-Oxum: 6376.2

% cat /tmp/RIPE-Atlas-Bolivia/manifest-sha256.txt
6467957fa9c06d30c1a72b62d13a224a3cdb570e5f550ea1d292c09f2293b35d  data/Makefile
3d65d66d6abcf1313ff7af7f94b7f591d2ad2c039bf7931701a936f1305ac728  data/bortzmeyer-ripe-atlas-lapaz.t
```

Les condensats ont été faits avec SHA-256.

Le fichier obligatoire `bagit.txt` doit indiquer le numéro de version et l'encodage. Le RFC décrit la version 1.0 mais, comme vous pouvez le voir plus haut, j'ai utilisé un outil un peu ancien pour fabriquer le sac. Le répertoire `data/` contient les fichiers de données, non modifiés (BagIt les traite comme du contenu binaire, copié au bit près). `manifest-sha256.txt` contient une ligne par fichier de données, indiquant le condensat. Notez qu'il peut y avoir plusieurs manifestes, avec des algorithmes différents. Cela permet, si un nouvel algorithme de condensation plus solide apparait, d'ajouter le manifeste au sac. Les noms d'algorithmes de condensation sont tirés du registre IANA <https://www.iana.org/assignments/named-information/named-information.xml#hash-alg> du RFC 6920 [1]. Quant aux fichiers (non obligatoires) dont le nom commence par `tagmanifest`, ils indiquent les condensats des fichiers de métadonnées :

```
% cat /tmp/RIPE-Atlas-Bolivia/tagmanifest-sha256.txt
16ed27c2c457038ca57536956a4431de4ac2079a7ec8042bab994696eb017b90 manifest-sha512.txt
b7e3c4230ebd4d3b878f6ab6879a90067ef791f1a5cb9ffc8a9cb1f66a744313 manifest-sha256.txt
91ca8ae505de9266e37a0017379592eb44ff0a2b33b240e0b6e4f2e266688a98 bag-info.txt
e91f941be5973ff71f1dccbdd1a32d598881893a7f21be516aca743da38b1689 bagit.txt
```

Enfin, le facultatif `bag-info.txt` contient des métadonnées qui ne sont typiquement prévues que pour les humains, pas pour être analysées automatiquement. La syntaxe est la classique `Nom: Valeur`. Certains des noms sont officiellement réservés (`Contact-Name`, `Bagging-Date`...) et on peut en ajouter d'autres à volonté.

Le sac est un répertoire, pas un fichier, et ne peut donc pas être transporté simplement, par exemple avec le protocole HTTP. On peut utiliser rsync, ou bien le sérialiser, par exemple en zip.

Il est amusant de noter qu'un sac peut être incomplet : des fichiers de données peuvent être stockés à l'extérieur, et récupérés dynamiquement lorsqu'on vérifie l'intégrité du sac. Dans ce cas, le condensat dans le manifeste permettra de vérifier qu'on a bien récupéré le contenu attendu. Les URL où récupérer ce contenu supplémentaire seront dans un fichier `fetch.txt`.

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc6920.txt`

Un sac peut être **complet** (ou pas) et ensuite **valide** (ou pas). La section 3 de notre RFC définit ces termes : un sac est complet s'il contient tous les fichiers obligatoires, et que tous les fichiers dans les manifestes sont présents, et que les fichiers comme `bagit.txt` ont une syntaxe correcte. Un sac est valide s'il est complet **et que** tous les condensats sont corrects.

La section 5 du RFC détaille quelques questions de sécurité liées à BagIt :

— BagIt va fonctionner entre systèmes d'exploitation différents. Les noms de fichiers peuvent comporter des caractères qui sont spéciaux pour un système d'exploitation mais pas pour un autre. Une mise en œuvre de BagIt sur Unix, par exemple, ne doit donc pas accéder aveuglément à un fichier commençant par une barre oblique, ou bien comprenant `../../../`. (Une suite de tests <https://github.com/LibraryOfCongress/bagit-conformance-suite> existe, avec plusieurs sacs. . .intéressants, permettant de tester la robustesse d'une mise en œuvre de BagIt.)

— La possibilité de récupérer des fichiers distants ouvre évidemment plein de questions de sécurité amusantes (les plus évidentes sont résolues par l'utilisation de condensats dans les manifestes, je vous laisse chercher les autres).

Sans que cela soit forcément un problème de sécurité, d'autres différences entre systèmes d'exploitation peuvent créer des surprises (section 6 du RFC), par exemple l'insensibilité à la casse de certains systèmes de fichiers, ou bien la normalisation Unicode. La section 6 est d'ailleurs une lecture intéressante sur les systèmes de fichiers, et leurs comportements variés.

Passons maintenant aux programmes disponibles. Il en existe en plusieurs langages de programmation. Le plus répandu semble bagit-python <https://github.com/LibraryOfCongress/bagit-python>, en Python. Il a été développé à la bibliothèque du Congrès, un gros utilisateur et promoteur de BagIt. La documentation <http://libraryofcongress.github.io/bagit-python/> est simple et lisible. On installe d'abord :

```
% pip3 install bagit
Collecting bagit
  Downloading https://files.pythonhosted.org/packages/ee/11/7a7fa81c0d43fb4d449d418eba57fc6c77959754c5c2259a2151
Building wheels for collected packages: bagit
  Running setup.py bdist_wheel for bagit ... done
  Stored in directory: /home/bortzmeyer/.cache/pip/wheels/8d/77/f7/8f91043ef3c99bbab558f578d19ce5938896e37e57609
Successfully built bagit
Installing collected packages: bagit
Successfully installed bagit-1.7.0
```

On peut ensuite utiliser cette bibliothèque depuis Python :

```
% python3
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import bagit
>>> bag = bagit.make_bag('/tmp/toto', {'Contact-Name': 'Ed Summers'})
>>>
```

Le répertoire `/tmp/toto` aura été transformé en sac.

Cette bibliothèque vient aussi avec un outil en ligne de commande. J'ai créé le premier sac d'exemple de cet article avec :

```
% bagit.py --contact-name 'Stéphane Bortzmeyer' --contact-email 'stephane+atlas@bortzmeyer.org' \
            /tmp/RIPE-Atlas-Bolivia
```

Et ce même outil permet de vérifier qu'un sac est valide :

```
% bagit.py --validate /tmp/RIPE-Atlas-Bolivia
2018-12-20 16:16:09,700 - INFO - Verifying checksum for file /tmp/RIPE-Atlas-Bolivia/data/bortzmeyer-ripe-at
2018-12-20 16:16:09,700 - INFO - Verifying checksum for file /tmp/RIPE-Atlas-Bolivia/data/Makefile
2018-12-20 16:16:09,701 - INFO - Verifying checksum for file /tmp/RIPE-Atlas-Bolivia/manifest-sha256.txt
2018-12-20 16:16:09,701 - INFO - Verifying checksum for file /tmp/RIPE-Atlas-Bolivia/manifest-sha512.txt
2018-12-20 16:16:09,701 - INFO - Verifying checksum for file /tmp/RIPE-Atlas-Bolivia/bagit.txt
2018-12-20 16:16:09,701 - INFO - Verifying checksum for file /tmp/RIPE-Atlas-Bolivia/bag-info.txt
2018-12-20 16:16:09,702 - INFO - /tmp/RIPE-Atlas-Bolivia is valid
```

Il existe d'autres mises en œuvre comme bagit `<https://godoc.org/github.com/ndlib/bendo/bagit>` ou bagins `<https://github.com/APTrust/bagins>` en Go. L'article « *"Using BagIt in 2018"* `<https://patchbaytech.wordpress.com/2017/12/15/using-bagit-in-2018/>` » donne des informations utiles dans d'autres langages.