Backup

Prérequis

- Une clé **GPG** pour le cryptage des données ;
- Un compte **hubiC** pour la sauvegarde à distance ;
- La liste des fichiers à sauvegardé.

Installation et configuration

hubiC

La version Linux étant encore en bêta, nous récupérons le package (.deb) de la dernière version en date directement depuis le serveur OVH [1)] avec un wget puis on procède à son installation.

```
# wget
http://mir7.ovh.net/ovh-applications/hubic/hubiC-Linux/2.1.0/hubiC-Linux-2.1
.0.53-linux.deb
# dpkg -i hubiC-Linux-2.1.0.53-linux.deb
```

On obtiendra alors des erreurs à propos de dépendances non satisfaites :

```
Selecting previously unselected package hubic.
(Reading database ... 105079 files and directories currently installed.)
Unpacking hubic (from hubiC-Linux-2.1.0.53-linux.deb) ...
dpkg: dependency problems prevent configuration of hubic:
 hubic depends on mono-runtime (>= 2.10.1); however:
  Package mono-runtime is not installed.
 hubic depends on libmono-posix4.0-cil (>= 2.10.1); however:
  Package libmono-posix4.0-cil is not installed.
 hubic depends on libmono-sqlite4.0-cil (>= 2.10.1); however:
  Package libmono-sqlite4.0-cil is not installed.
 hubic depends on libmono-system4.0-cil (>= 2.10.1); however:
  Package libmono-system4.0-cil is not installed.
 hubic depends on libmono-system-configuration4.0-cil (>= 2.10.1); however:
  Package libmono-system-configuration4.0-cil is not installed.
 hubic depends on libmono-system-core4.0-cil (>= 2.10.1); however:
  Package libmono-system-core4.0-cil is not installed.
 hubic depends on libmono-system-data4.0-cil (>= 2.10.1); however:
  Package libmono-system-data4.0-cil is not installed.
 hubic depends on libmono-system-data-datasetextensions4.0-cil (>= 2.10.1); however:
  Package libmono-system-data-datasetextensions4.0-cil is not installed.

dpkg: error processing hubic (--install):
 dependency problems - leaving unconfigured
Processing triggers for man-db ...
Processing triggers for hicolor-icon-theme ...
Errors were encountered while processing:
 hubic
```

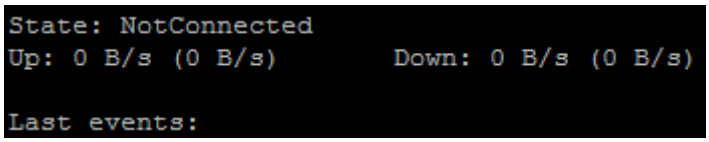
On corrigera ces erreurs via la commande suivante qui se chargera d'installer tout le nécessaire.

```
# apt-get install -f
```

A ce stade, si l'on essaye de lancer l'application, nous obtiendrons l'erreur suivante : *Cannot contact daemon, are you sure it is running?*
La documentation du service nous indique que ce problème survient sur les serveurs ou lors de l'utilisation d'une connexion SSH. On va donc reconfigurer dbus correctement :

```
# export DBUS_SESSION_BUS_ADDRESS=$(dbus-daemon --session --fork --print-address)
```

hubiC est maintenant prêt !



GPG

Nous n'avons ici aucun paquet à installé, tout étant déjà présent par défaut au sein de Linux. Nous aurons juste à importer notre clé GPG **publique**.

Pour se faire, nous créons un fichier texte, *gpg.txt* par exemple, et y collons notre clé publique. Ne reste qu'à l'importé via


```
# gpg --import gpg.txt
```

Mise en place

Principe

Nous utiliserons la fonction d'archivage incrémentiel de **tar** pour créer nos sauvegardes.
Nous allons donc créer un dossier, nous avons choisi de le mettre à la racine de notre système et le nommé *backups*. Nous y ferrons deux sous-dossiers : le premier, que nous appelons *temp* servira à stocker nos archives en cours de traitement ; le second, *save* sera indiquer à hubiC en mode synchronisation pour l'envoie vers le service.

Passons à la création de l'archive :

```
tar -czf /backups/temp/nomArchive_`date --rfc-3339=date`.tar.gz --listed-incremental=/backups/nomArchive.list /unChemin/*
```

nomArchive correspond au nom de l'archive souhaiter auquel nous ajoutons la date au format AAAA-MM-JJ. *nomArchive.list* contient la liste des fichiers et de leur état depuis la dernière fois que nous avons exécuter l'opération.

Une fois l'archive créée, on la crypte pour assurer la confidentialité de nos données.

```
gpg --encrypt --recipient idKeyPgp nomArchive.tar.gz
```

idKeyPgp étant l'id de votre clé publique précédement importer et *nomArchive*, l'archive créer précédemment.

Enfin, on déplace notre archive gpg (*nomArchive.tar.gz.gpg*) vers le dossier *save* en vue de sa synchronisation sur **hubiC**, sans oublier de supprimer l'archive restante devenue inutile (*nomArchive.tar.gz*).

```
mv /backups/temp/nomArchive.tar.gz.gpg /backups/save/
rm /backups/temp/nomArchive.tar.gz
```

Script

On reproduit le principe sous forme de script dans lequel on précise les dossiers que l'on veut sauvegarder dans le tableau *bk_data*.

```
#!/bin/bash
#
# Création de sauvegarde
#

# Variables
bk_home="/backups"
bk_temp="/backups/temp"
bk_save="/backups/save"
declare -a bk_data=("/premier/dossier" "/deuxieme/dossier")

# Parcourir la liste
for folder in "${bk_data[@]}"
do
  echo "$folder - Debut de sauvegarde"

  # On se déplace dans le dossier
  cd $folder

  # Nom du répertoire à sauvegarder
  bk_name=`pwd | sed 's#.*/##'`

  # Creation de l'archive
  tar -czf $bk_temp/$bk_name"_"`date --rfc-3339=date`.tar.gz --listed-
incremental=$bk_home/$bk_name.list $folder

  # Cryptage de l'archive
```

```
  gpg --encrypt --recipient 2AC34D90 $bk_temp/$bk_name"_"`date --
rfc-3339=date`.tar.gz

  # Creation d'un sous-dossier si necessaire
  if [ ! -d "$DIRECTORY" ]; then
    mkdir $bk_save/$bk_name
  fi
```

```
  # Deplacer les fichiers vers le dossier de synchronisation hubiC
  mv $bk_temp/$bk_name"_"`date --rfc-3339=date`.tar.gz.gpg $bk_save/$bk_name
  rm $bk_temp/$bk_name"_"`date --rfc-3339=date`.tar.gz
  cp $bk_home/$bk_name.list $bk_save/$bk_name

  # Sortir du dossier à sauvegarder
  cd $bk_home

  echo "$folder - Fin de sauvegarde"
done
```

Il ne faudra pas oublier de lancer la synchronisation avec la commande

```
hubic login mailHubic dossierASynchro
```

On pourra également exécuter le script via une tache planifiée pour automatiser la tâche.

1)
https://forums.hubic.com/showthread.php?230-hubic-Linux-sortie-de-la-version-bêta