Module de type sync-regexp-tag

Sommaire

Concept

Il est possible de définir sur un Tagger une action qui va utiliser une plage d'adresses IP afin d'éditer automatiquement une propriété des hôtes, clusters, modèles d'hôtes et modèles de clusters issus de l'import des sources en fonction de la valeur d'une donnée ou d'une propriété.

- Cette action est portée par un module de type `sync-regexp-tag,` que vous aurez besoin d'ajouter sur votre Tagger.
- L'intérêt est, par exemple, d'ajouter le modèle d'hôte Bordeaux sur les hôtes dont le nom commence par "bdx" (*voir la page* Tagger utilisant un module basé sur des expressions régulières).

L'activation de ce module se fait en 3 étapes :

1. Définir le module de type `sync-regexp-tag`.
2. L'ajouter au Tagger qui exécutera cette règle.
    a. Si vous n'avez pas déjà de Tagger défini, il faut le mettre en place (*voir la page* Definition des taggers).
3. Redémarrer le Synchroniser pour que cette modification soit prise en compte.

Définition du module

La configuration des modules doit être placée dans le dossier **/etc/shinken/modules/**

- Un exemple de fichier de configuration est disponible ici : **/etc/shinken-user-example/configuration/daemons/synchronizers/modules/sync-regexp-tag/MY-MODULE-regexp-tag-example.cfg**
- Copier cet exemple et adapter le à votre besoin (*voir plus bas, la description des différents paramètres*).

```
cp /etc/shinken-user-example/configuration/daemons/synchronizers/modules/sync-regexp-tag/MY-MODULE-regexp-tag-example.cfg /etc/shinken/modules/MY-MODULE-regexp-tag.cfg
```

Nous vous recommandons d'intégrer, dans le nom du module, un rappel du type du module. C'est plus simple en gestion (*ex : MY-MODULE-regexp-tag.cfg*).

- Renommer le module (*paramètre* ***module_name***).
- Éditer les paramètres pour définir l'action du module :
    - Les hôtes à impacter.
    - La nature de l'action.

Exemple de fichier de configuration

Cet exemple ajoute

- le modèle d'hôte Bordeaux, à la fin des modèles d'hôtes utilisés par chaque hôte
- dont le nom commence par : bdx

```
#================================================================================
# ip-tag-*
#================================================================================
# Daemons that can load this module:
# - synchronizer (into a tagger object)
# This module add new templates into hosts objects based on their hostname (regexp check)
#================================================================================


define module{

    #======== Module identity =========
    # Module name. Must be unique
    module_name     MY-MODULE-regexp-tag

    # Module type (to load module code). Do not edit.
    module_type     sync-regexp-tag

    #======== Regexp definition and objects edition =========
    # matched_prop: which object property to check the regexp
    matched_prop    host_name

    # regexp to try to detect
    matched_regexp  ^bdx.*

    # property: which property to edit on the hosts. Default: use   (templates definitions)
    property               use

    # method: how to edit the host "property". Several methods are available:
    # - replace = put the value if not another one is in place
    # - append  = add the value at the END
    # - prepend = add the value at the BEGINING
    # - set     = just the value, erase the previous value set by other tagger or source.
    method                 append

    # value:  which value to set on the property
    value                  Bordeaux

}
```

Détails des sections composant le fichier de configuration

Identification du module

```
#======== Module identity =========
# Module name. Must be unique
module_name     MY-MODULE-regexp-tag

# Module type (to load module code). Do not edit.
module_type     sync-regexp-tag
```

Il est possible de définir plusieurs instances de module `sync-regexp-tag` dans l'architecture Shinken.

- Chaque instance devra avoir un nom unique.

Nom	Type	Unité	Défaut	Description

`module_name`	Texte	---	**---**	**Valeur obligatoire**. Nom du Module. • Doit être unique parmi tous les modules.
`module_type`	Texte	---	**sync-regexp-tag**	Ne pas modifier.

Définition des hôtes où l'action de modification sera appliquée (expression régulière)

```
#======== Regexp definition and objects edition =========
# matched_prop: which object property to check the regexp
matched_prop    host_name

# regexp to try to detect
matched_regexp  ^bdx.*
```

Nom	Type	Unité	Défaut	Description
`matched_prop`	Texte	Nom de propriété d'un hôte	**---**	**Valeur obligatoire** La propriété comparée à l'expression régulière.
`matched_regexp`	Texte	regexp python	**---**	**Valeur obligatoire** Regexp auquel la propriété doit correspondre.

Ecrire une Expression Régulière (*Regexp*) pour la propriété matched_regexp

Une expression régulière est une séquence de caractères qui forme un motif de recherche.

- Ce motif est utilisé pour rechercher des correspondances dans des textes.
- Les caractères simples, comme `a`, `b`, `1`, `2`, etc., recherchent une correspondance exacte.

Par exemple, le motif "`bdx`" recherchera exactement la séquence "bdx" dans la propriété ou la donnée définie dans "matched_prop". Les valeurs ci-dessous contiennent le motif "bdx", elles correspondent donc au motif défini et le Tagger appliquera la modification :

- **bdx**-serveur-mysql
- serveur-mysql-**bdx**
- serveux-**bdx**-mysql

Certains caractères sont spéciaux et ont une signification particulière dans les expressions régulières. Par exemple :

- `.` : Correspond à n'importe quel caractère sauf un retour à la ligne.
- `*` : Correspond à zéro ou plusieurs occurrences du caractère précédent.
- `+` : Correspond à une ou plusieurs occurrences du caractère précédent.
- `?` : Correspond à zéro ou une occurrence du caractère précédent.
- `[]` : Correspond à n'importe quel caractère à l'intérieur des crochets. Par exemple, `[abc]` cherchera les lettres "a", "b" ou "c".
- `()` : Permet de grouper des motifs.
- `|` : Correspond à l'un ou l'autre des motifs séparés par le pipe.
  - Par exemple, le motif "`(bdx|bordeaux)`" cherchera dans un texte la chaîne "bdx" ou "bordeaux".
    - Si au moins une des deux est présente, alors la chaîne correspond au motif.
- `^` : Correspond au début d'une ligne.
- `$` : Correspond à la fin d'une ligne.

Pour chercher un de ces caractères spéciaux dans un texte (*par exemple si le nom de l'hôte contient le signe dollar $*) il faut le faire précéder d'un antislash, comme ceci : \$

- Ces caractères servent à affiner la recherche.

- Pour plus d'explications sur comment écrire des expressions régulières, il est possible de se référer à la documentation de Python (*voir la page* Guide des expressions régulières).

Exemples

Si seuls les éléments dont le nom commence par "bdx" doivent recevoir le modèle d'hôte Bordeaux, le motif précédent ne suffit plus, pour cela il faut également utiliser le symbole ^ :

```
^bdx
```

Si le nom doit **commencer** par "bdx" **OU** "bordeaux" il faut également ajouter les parenthèses :

```
^(bdx|bordeaux)
```

Exemples de valeurs :

Texte	Correspondance avec le motif
bdx-serveur-1	**OUI**
serveur-mysql1-bdx	**NON** car "bdx" n'est pas au début
bordeaux-mysql	**OUI**
Bordeaux-mysql	**NON** car le B est une lettre majuscule

Édition des hôtes

```
# property: which property to edit on the hosts. Default: use   (templates definitions)
property              use

# method: how to edit the host "property". Several methods are available:
# - replace = put the value if not another one is in place
# - append  = add the value at the END
# - prepend = add the value at the BEGINING
# - set     = just the value, erase the previous value set by other tagger or source.
method                append

# value:  which value to set on the property
value                 Bordeaux
```

Nom	Type	Unité	Défaut	Description
`property`	Texte	---	**---**	**Valeur obligatoire** Nom de la propriété à modifier. Ex : use pour les modèles de l'hôte.
`method`	Texte	---	**append**	**Valeur obligatoire** Méthode d'édition : • **replace** : ajoute la valeur si elle est absente. • **append** : ajoute la valeur à la fin. • **prepend** : ajoute la valeur au début. • **set** : remplace la valeur actuelle ou l'ajoute si absente.
`value`	Texte	---	**---**	**Valeur obligatoire** Valeur utilisée par la méthode.

Déclarer le Module dans le Tagger

Pour qu'un Tagger utilise les règles définies dans un module, il faut ajouter le nom du module au paramètre `modules` dans le fichier de configuration du Tagger (*ex : /etc/shinken/taggers/MY-TAGGER.cfg*)

Exemple avec un Tagger appelé MY-TAGGER :

```
define tagger {

    tagger_name     MY-TAGGER
    order           1

    modules         TAGGER-MODULE-01, TAGGER-MODULE-02, MY-MODULE-regexp-tag

    description
}
```

Pour prendre en compte le changement de configuration, il faut redémarrer le Synchronizer :

```
service-shinken-synchronizer restart
```

Affichage des Taggers sur l'interface de Configuration

Les règles apportées par chaque module apparaissent à l'intérieur du Tagger, dans l'onglet **Règles** :