Module de type sync-regexp-tag

Sommaire

Concept

Un Tagger basé sur des expressions régulières s'applique les éléments suivants issus de l'import des sources :

- les hôtes,
- les clusters,
- les modèles d'hôtes ou
- les modèles de clusters.

Il utilise une expression régulière (*regexp*) sur une propriété ou une donnée pour sélectionner les éléments auxquels appliquer la modification configurée pour le Tagger.

Un cas d'usage classique est de rajouter/modifier des modèles d'hôtes/clusters sur les éléments s'ils respectent une règle de nommage sur le nom (*voir la page Tagger basé sur des expressions régulières*).

L'activation d'un Tagger se fait en trois étapes :

1. Définir un Tagger qui utilise un module de type sync_regexp_tag (*ex : /etc/shinken/taggers/regexp-tags.cfg*).
2. Configurer les règles du module (*ex : /etc/shinken/modules/sync-regexp-tag.cfg*).
3. Déclarer le Tagger dans le Synchronizer (*/etc/shinken/synchronizers/synchronizer-master.cfg*).

Définition du tagger

La configuration des Taggers doit être placée dans le dossier **/etc/shinken/taggers/.**

Le Tagger basé sur des expressions complexes livré par Shinken : **/etc/shinken/taggers/regexp-tags.cfg**.

Un exemple de fichier de configuration est disponible : **/etc/shinken-user-example/configuration/daemons/synchronizers/taggers/regexp-tags/regexp-tags.cfg**

Exemple de fichier de configuration

```
define tagger {
# Shinken Enterprise. Lines added by import core. Do not remove it, it's used by Shinken Enterprise to
update your objects if you re-import them.
    _SE_UUID            core-tagger-f69a7f125adc11e5ae66080027f08538
    _SE_UUID_HASH       d183bc0dd8dba3d7a98725f676d7283c
# End of Shinken Enterprise part


    tagger_name         regexp-tags
    order               1

    modules             sync-regexp-tag


    description         This tagger will tag host based on the host_name
}
```

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

```
service shinken-synchronizer restart
```

Détails des sections composant le fichier de configuration

Il est possible de définir plusieurs instances de Taggers dans l'architecture Shinken.

Chaque instance devra avoir un nom unique.

Nom	Type	Unité	Défaut	Commentaire
`tagger_name`	Texte	---	**---**	**Valeur obligatoire** Nom du Tagger. Le paramètre doit être unique parmi les autres taggers. Caractères interdits : les signes inférieur ou supérieur (*< ou >*), les guillemets (*' ou "*).
`order`	Entier	---	**---**	**Valeur obligatoire** L'ordre d'application des Taggers. Les Taggers sont exécutés les uns à la suite des autres dans l'ordre croissant de la propriété order. Si plusieurs Taggers qui s'enchaînent modifient la même propriété, il est possible qu'un Tagger annule les modifications de précédents Taggers. Attention donc à l'ordre d'exécution des Taggers
`modules`	Texte	---	**---**	**Valeur obligatoire** Nom du module utilisé par le Tagger.
`description`	Texte	---	**---**	Une description du Tagger affichée dans l'Interface de Configuration.

Configuration du module

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

La configuration du module livré par Shinken se trouve dans le fichier **/etc/shinken/modules/sync-regexp-tag.cfg**

Un exemple de fichier de configuration est disponible ici **: /etc/shinken-user-example/configuration/daemons/synchronizers/modules/sync-regexp-tag/sync-regexp-tag-example.cfg**

Exemple de fichier de configuration

```
#====================================================================================
# 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{

# Shinken Enterprise. Lines added by import core. Do not remove it, it's used by Shinken Enterprise to
update your objects if you re-import them.
    _SE_UUID             core-module-be8578085adb11e59c43080027f08538
    _SE_UUID_HASH        05673d95f7f69b0c1c291e8343b6ec93
# End of Shinken Enterprise part

    #======== Module identity =========
    # Module name. Must be unique
    module_name     sync-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  .*shinkendemo.*

    # 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                 SHINKEN-DEMO-REGEXP-EXAMPLE
}
```

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

```
service shinken-synchronizer restart
```

Détails des sections composant le fichier de configuration

Identification du module

```
# Shinken Enterprise. Lines added by import core. Do not remove it, it's used by Shinken Enterprise to
update your objects if you re-import them.
    _SE_UUID            core-module-be8578085adb11e59c43080027f08538
    _SE_UUID_HASH       05673d95f7f69b0c1c291e8343b6ec93
# End of Shinken Enterprise part

    #======== Module identity =========
    # Module name. Must be unique
    module_name     sync-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 de type `sync-regexp-tag` dans l'architecture Shinken.

Chaque instance devra avoir un nom unique.

Nom	Type	Unité	Défaut	Commentaire
`module_name`	Texte	---	**---**	**Valeur obligatoire** Nom du module. Doit être unique parmi les modules.
`module_type`	Texte	---	**sync-regexp-tag**	Ne pas modifier.

Définition de l'expression régulière et de l'édition d'objets

```
...
    #======== 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  .*shinkendemo.*

    # 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                 SHINKEN-DEMO-REGEXP-EXAMPLE
...
```

Nom	Type	Unité	Défaut	Commentaire
`matched_prop`	Texte	Nom de propriété d'un hôte	**host_name**	**Valeur obligatoire** La propriété comparée à l'expression régulière.

`matched_regexp`	Texte	regexp python	**---**	**Valeur obligatoire** Regexp auquel la propriété doit correspondre.
`property`	Texte	Nom de propriété d'un hôte	**---**	**Valeur obligatoire** Nom de la propriété à modifier.
`method`	Texte	---	**replace**	**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.

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 principalement 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 caractères.
- `|` : Correspond à l'un ou l'autre des motifs séparés par le pipe. Par exemple, "`(bdx|bordeaux)`" correspond à "bdx" ou "bordeaux".
- `^` : 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

Déclarer le Tagger dans le Synchronizer

L'activation des Taggers se fait dans le fichier de configuration du Synchronizer

- Il faut ajouter le nom du Tagger au paramètre `taggers` dans le fichier de configuration du démon Synchronizer (*/etc/shinken/synchronizers/synchronizer-master.cfg*) .

Exemple : Le Tagger livré par défaut dont le nom est "regexp-tags" :

```
define synchronizer {
    [...]
    #======== Taggers =========
    taggers                  TAGGER_01, regexp-tags
    [...]
}
```

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 Taggers apparaissent sur la page Principale de l'Interface de Configuration (*voir la page* Page Principale).