Configuration de la Visualisation de l'architecture

Sommaire

Introduction

La génération des vues de l'architecture est de la responsabilité de l'addon "**nagvis-shinken-architecture**".

Les sections suivantes décrivent son comportement, comment l'activer/désactiver et les différents paramètres de configuration disponibles.

Activation et désactivation de l'addon

Activation automatique

Lors d'une installation ou d'une mise à jour depuis une version antérieure à la V02.05.00, l'addon est automatiquement installé. Son activation dépend ensuite du type de machine sur laquelle Shinken est installé.

- L'addon "**nagvis-shinken-architecture**" analyse la configuration reçue par l'Arbiter pour ensuite générer les vues graphiques des royaumes et les hôtes correspondants.
- Pour cette raison, activer cet addon sur une machine où aucun démon Arbiter n'est actif n'a pas de sens.
  - Sur une machine n'ayant qu'un démon Poller actif par exemple, l'addon ne sera donc pas activé.
  - Sur la machine principale d'une installation Shinken (*c'est-à-dire celle qui contient l'Arbiter*), l'addon "**nagvis-shinken-architecture**" sera automatiquement activé lors d'une installation ou mise à jour depuis une version antérieure à la V02.05.00.

Activer/désactiver la fonctionnalité

Si cette fonctionnalité ne correspond pas à un besoin dans votre utilisation de Shinken Entreprise, elle peut être désactivée et réactivée selon les envies.

- Les sections suivantes décrivent de manière succincte les différentes commandes permettant de manipuler les addons.
- Plus de détails sur ces commandes se trouvent dans la page de documentation dédiée : Activation - désactivation des outils supplémentaires (addons)

Vérifier l'état d'activation des addons

La commande "**shinken-addons-list**" permet de lister les addons ainsi que leur état d'activation. Dans l'exemple, on voit que l'addon "**nagvis-shinken-architecture**" est désactivé.

Activer l'addon

Les addons peuvent être activés avec la commande "**shinken-addons-enable**". Par exemple, pour activer l'addon "**nagvis-shinken-architecture"**, la commande est la suivante:

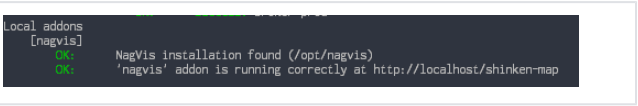
```
shinken-addons-enable nagvis-shinken-architecture
```

Désactiver l'addon

Les addons peuvent être désactivés avec la commende "**shinken-addons-disable**". Avec le même exemple que précédemment, la commande est la suivante:

```
shinken-addons-disable nagvis-shinken-architecture
```

Visualiser l'état de l'addon



L'état général de fonctionnement de l'addon peut être vérifié avec la commande **shinken-healthcheck**.

Dans le Healthcheck, l'état de l'addon est visible dans la section "Addons". Les points suivants sont vérifiés pour l'addon "'**nagvis-shinken-architecture**" :

- Présence du fichier de configuration correspondant (*/etc /shinken/module/architecture-export.cfg*)
- Présence de l'installation NagVis (*composant responsable de l'affichage des architectures*)
- L'interface d'affichage des informations de l'architecture est joignable.

Configuration de l'addon

Manipulation et étapes de configuration

L'activation et désactivation de la fonctionnalité de visualisation de l'architecture Shinken s'effectue via l'addon "**nagvis-shinken-architecture**".

- La configuration de cet addon s'effectue via le fichier de configuration du module "**architecture-export**", situé dans " ***/etc/shinken/modules /architecture-export.cfg*** " (*voir Module architecture-export*).

Fonctionnement du module

L'export de l'architecture s'effectue grâce au module "**architecture-export**". Le fonctionnement de ce module est séparé en 2 parties :

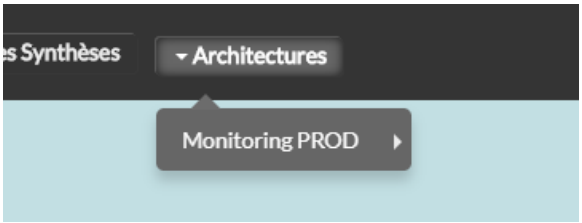
- **Envoi** : Lorsqu'un changement d'architecture est détecté, le module envoie la description de l'architecture de l'installation Shinken sur laquelle il est installé à une liste de modules destinataires.
- **Réception** : Le module reste également en attente de données. Lorsqu'il reçoit des données correspondant à une description d'architecture, il génère les vues détaillées des royaumes et l'arbre des royaumes correspondants.

Dans le module, ces deux aspects peuvent donc être configurés.

Aussi, le module est positionné sur l'Arbiter. On aura donc un seul module "**architecture-export**" par installation Shinken.

Différents paramètres de configuration

Nom de l'architecture



Lorsque le module "**architecture-export**" reçoit une description d'architecture, il faut pouvoir identifier l'architecture correspondante. Le paramètre **architecture_name** permet donc de spécifier le nom de l'architecture qui sera visible dans l'interface de visualisation.

Ce nom est également présent dans les hôtes générés et donc également dans les cartes NagVis.

Lors d'une installation Shinken, ce nom est défini par défaut à :

> Shinken-<hostname_machine_arbiter>

/etc/shinken/modules/architecture-export.cfg

```
define module {
    #======== Module identity =========
    # Module name. Must be unique
    module_name             architecture-export

    # Module type (to load module code). Do not edit.
    module_type             architecture_export


    .
    .
    .
    (contenu)
    .
    .
    .

    # Name with which this Shinken installation will be identified in the NagVis maps
    # The following characters are forbidden in the architecture name: ~!$%^&*"'|<>?,()=/+
    architecture_name       Monitoring PROD

    .
    .
    .
    (suite du fichier)
}
```

Port et interface d'écoute du module

Lors d'un changement de l'architecture, la description de cette architecture est envoyée à un ou plusieurs autres modules. Il est possible de modifier les paramètres d'écoute du module, comme le port et l'interface d'écoute :

/etc/shinken/modules/architecture-export.cfg

```
define module {
    .
    .
    .
        (contenu)
    .
    .
    .

    #======== Architecture description communication =========
    # This module opens a listening socket on which other shinken installations will send their architecture
description.
    # When an architecture description is received by the module, it creates corresponding hosts and NagVis
maps.


    # host: interface used for the listening socket (0.0.0.0 = all interfaces)
    host                    0.0.0.0


    # Port to use for the listening socket
    port                    7780

    .
    .
    .
    (suite du fichier)
}
```

Sécurisation de la communication (SSL)

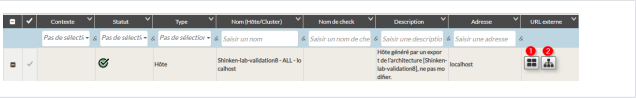
Il est également possible de forcer la connexion au module en HTTPS. Pour activer une communication sécurisée, il faut positionner le paramètre **use_ssl** à 1 (*par défaut 0*).

Pour utiliser son propre certificat, il faudra le spécifier avec les paramètres **ssl_cert** et **ssl_key**. Par défaut, le certificat livré avec Shinken est utilisé.

/etc/shinken/modules/architecture-export.cfg

```
define module {
    ...
    # 0 = Use HTTP, 1 = Use HTTPS
    use_ssl                 0
    ssl_cert                /etc/shinken/certs/server.cert
    ssl_key                 /etc/shinken/certs/server.key
    ...
}
```

URL d'accès dans la visualisation



Dans l'interface de Visualisation, des liens sont présents dans la barre supérieure permettant un accès rapide à NagVis qui affiche l'architecture.

- Cette adresse est construite en fonction de l'adresse du démon Arbiter.
- Ces liens sont également générés et ajoutés à la "**Liste des URL externes**" (*notes_multi_url*) de l'hôte qui héberge le démon Arbiter par ce module.

Vous aurez donc un lien permettant d'accéder à la vue détaillée de votre architecture **(1)** et à l'arbre de vos royaumes **(2)**.

Dans certains cas, il se peut que cette adresse soit incorrecte. C'est le cas par exemple si l'adresse de l'Arbiter est l'adresse d'une interface locale utilisée pour la communication entre démons, tandis que NagVis et les interfaces de Configuration et Visualisation sont présentées sur une interface publique.

Dans le fichier de configuration du module, le paramètre map_base_url permet de spécifier la base de l'adresse utilisée pour la construction des adresses d'accès aux cartes. Pour être fonctionnelle, cette adresse doit correspondre à l'adresse de la machine qui héberge l'Arbiter.

/etc/shinken/modules/architecture-export.cfg

```
define module {
    ...
    # Base of URL used to display links in the Visualization UI
    # Defaults to Arbiter URL if empty
    # map_base_url  http://example.com/
    ...
}
```

Ce paramètre peut être utilisé pour présenter un lien HTTPS vers NagVis (*si NagVis a été configuré pour être servi en HTTPS dans Apache*).

Par exemple :

```
map_base_url  https://example.com/
```

Liste des destinataires

Enfin, lorsqu'un changement d'architecture est détecté, la description de l'architecture est envoyée à une liste de destinataires. Par défaut, le module "**architecture-export**" s'envoie les informations de l'architecture pour générer l'arbre des royaumes et les vues détaillées, mais il est également possible d'envoyer l'architecture à d'autres modules "**architecture-export**". Il faudra, dans la liste des destinataires, prendre en compte le port ainsi que l'utilisation de SSL.

/etc/shinken/modules/architecture-export.cfg

```
define module {
    ...
    # Architecture description recipients
    #   When the architecture of this Shinken installation changes (realms and daemons configuration),
    #   and the arbiter is restarted the architecture description will be sent to the following hosts.
    send_my_architecture_to_recipients   http://127.0.0.1:7780,https://addressemodule2:1234
    ...
}
```

Il n'est pas non plus obligatoire d'envoyer la description de l'architecture sur 127.0.0.1. Si on omet cette adresse dans la liste des destinataires, les vues détaillées et l'arbre des royaumes ne seront pas générées sur l'installation Shinken actuelle, mais seulement sur celles spécifiées dans la liste des destinataires.

(*Pour un cas d'application concret, voir la page* Exemple pratique - Mise en place automatisée d'une plateforme de supervision secondaire)

Communiquer avec Graphite

Dans le cas où vous avez choisi une installation distribuée pour Graphite, l'architecture export doit être configurée de sorte à le détecter. Pour cela, il faut établir une communication avec la machine où Graphite est installée afin que l'architecture export puisse créer les hôtes associés à celle-ci, y ajouter les checks correspondants et l'afficher sur la carte NagVis.
Il est donc possible d'ajouter quatre paramètres spécifiant les différentes informations de connexion en SSH à votre machine.

```
define module {
    ...
    # SSH Settings for graphite hosts discovery
    ssh_port                            22
    ssh_user                            shinken
    ssh_key_file                        /var/lib/shinken/.ssh/id_rsa.pub
    ssh_timeout                         5
    ...
}
```

Dans le cas où ces paramètres ne sont pas renseignés, il sera impossible pour l'architecture export de recenser l'architecture Graphite.

Trier les royaumes

Les royaumes peuvent être affichés soit par ordre alphabétique, soit par taille.

Pour cela, il faut ajouter le paramètre map_realm_layout dans :

/etc/shinken/modules/architecture-export.cfg

```
define module {
    ...
    # Sort order for realms in the NagVis maps
    # - sort_by_name
    # - sort_by_size (default)
    map_realm_layout sort_by_name
    ...
}
```

Si le paramètre n'est pas renseigné dans le fichier de configuration, c'est l'affichage par taille qui sera pris en compte par défaut.

Communication entre Nagvis et Shinken

Afin d'afficher les statuts des objets Shinken et de correctement rediriger vers la WebUI en cas de clic sur un objet, NagVis doit pouvoir communiquer avec deux modules du Broker : **Livestatus** et **WebUI**.

À l'installation, NagVis communiquera avec les modules du broker par défaut à savoir le **broker-master**.

Il vous est cependant possible de modifier ceci via quatre paramètres du module "**architecture-export**".

nom	type	défaut	commentaire
`architecture_export__broker_connection__broker_name`	Texte	**broker-master**	Nom du Broker sur lequel sont les modules WebUI et Livestatus avec lesquels NagVis communiquera
`architecture_export__broker_connection__broker_livestatus`	Texte	**Livestatus**	Nom du module de type Livestatus sur lequel NagVis récupérera les informations des éléments Shinken afin d'afficher leurs statuts sur les cartes.
`architecture_export__broker_connection__broker_webui_communication_type`	Texte	**WebUI**	Type de communication avec la WebUI souhaitée. Ce paramètre est utilisé pour la redirection lorsqu'on clique sur les liens de la carte. Valeurs possibles : • **module** (*si on souhaite spécifier le nom d'un module WebUI pour obtenir son adresse*) • **URL** (*si on souhaite renseigner une URL : l'adresse de la WebUI est derrière un reverse proxy ou utilise une adresse DNS*)
`architecture_export__broker_connection__broker_webui_target`	Texte	**Aucun**	WebUI avec laquelle communiquer. Si le paramètre précédent est à **module** celui-ci doit être le nom d'un module WebUI Sinon, ce doit être une URL redirigeant vers un module WebUI

/etc/shinken/modules/architecture-export.cfg

```
define module {
    ...
    #  #
    #      BROKER CONNECTION PARAMETERS      #
    #  #

    #  These parameters are used to allow NagVis to communicate with the Broker and modules you want
    #  Name of the Broker holding the modules you want NagVis to communicate with
    #
    #           Default : broker-master
    #
    # architecture_export__broker_connection__broker_name broker-master

    #  Name of the Livestatus module you want NagVis to communicate with to retrieve objects information
    #
    #           Default : Livestatus
    #
    # architecture_export__broker_connection__broker_livestatus Livestatus

    #  Type of the target WebUI you want to communicate with
    #  This allow redirection when clicking on object on the maps
    #
    #           Default : module => Use a WebUI module configuration to get it's address
    #           ...     : url    => Use a URL ( the WebUI address is behind a reverse proxy or use an DNS
    #                               address )
    #
    # architecture_export__broker_connection__broker_webui_communication_type module

    #  Targeted WebUI to communicate with
    #  If previous parameter is set to "module", this must be a WebUI name
    #  If previous parameter is set to "url", this must be a URL
    #
    #           Default : WebUI
    #
    # architecture_export__broker_connection__broker_webui_target WebUI
     ...
}
```