Pack Linux_by_ssh - NEW

Note : Si vous êtes intéressé par ce pack, veuillez nous contacter pour son téléchargement. Nous vous accompagnerons lors de l'installation de ce pack sur votre plateforme.

Sommaire

Contexte

Lorsque vous installez Shinken Enterprise, un certain nombre de modèles et de commandes sont inclus dans votre configuration.

Le pack "linux_by_ssh", comme son nom l'indique, permet de superviser des hôtes sur lesquels est installé un système d'exploitation basé sur Linux (*serveur ou client*).

Il contient 17 commandes, 17 modèles de checks dédiés à 2 modèles d'hôte spécifiques (*nommés "linux_by_ssh" et "linux_by_ssh_advanced"*).

Toutes les commandes de ce pack se basent sur des scripts présents dans le répertoire des scripts shinken **/var/lib/shinken/libexec** (ou **$PLUGINSDIR$** depuis l'interface de configuration).

Le protocole SSH (Secure Shell) est utilisé par chacun des 15 scripts du pack linux. Les scripts communiqueront avec votre machine directement par un invite de commande après s'être connectés avec les identifiants SSH que vous aurez paramétrés.

Nous allons ici détailler ces checks associés au modèle Linux de ce pack.

Sommaire des checks

Modèle Linux by SSH

Check Name	Description
CPU_Stats SSH	Récupère les statistiques du CPU
Disks Usage SSH	Récupère les informations de taille des disques
Load Average SSH	Récupère les informations de la charge système
Memory SSH	Récupère des informations concernant la RAM
NtpSync SSH	Récupère le délai/décalage avec le serveur NTP
Uptime SSH	Récupère le temps depuis le dernier démarrage

Modèle Linux by SSH avancé

Check Name	Description
Connection Failed SSH	Récupère et analyse les tentatives de connections échouées sur votre serveur
Disks Stats SSH	Récupère des informations supplémentaires des disques
Kernel Stats SSH	Récupère des informations du kernel
NET Stats SSH	Récupère des statistiques réseaux
Processes Memory SSH	Vérifie l'utilisation de la mémoire ram RSS (Resident Set Size) de chaque processus
NFS Stats SSH	Récupère des statistiques NFS
Read-only FileSystem SSH	Vérifie si un fichier système est en lecture seule

Security SSH	Vérifie les paramètres SSH de l'hôte et les compare avec ceux définis dans la configuration (via les données)
TCP States SSH	Récupère les états des ports TCP

Les modèles d'hôte et leurs données héritées

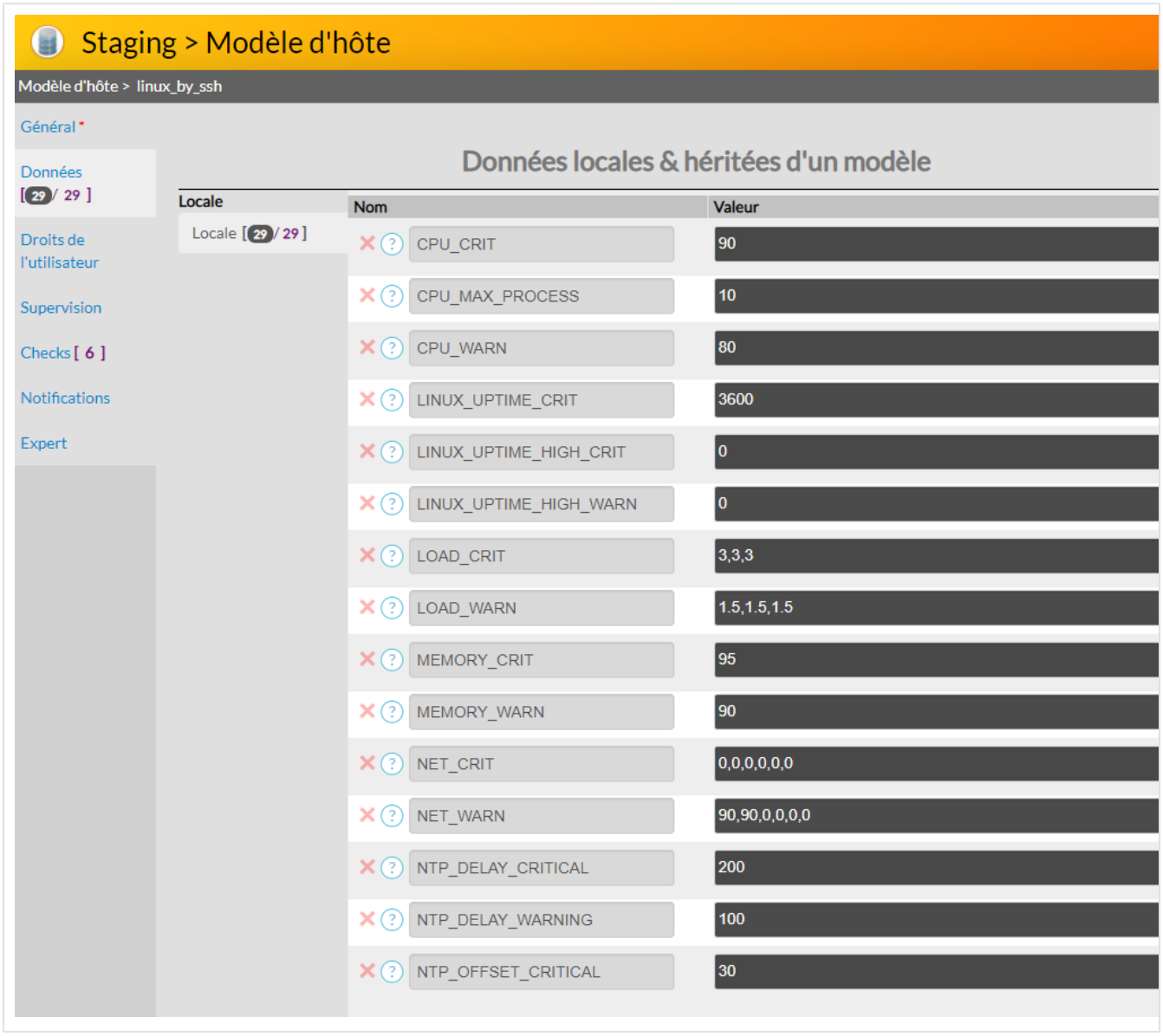
Les modèles d'hôtes "linux_by_ssh" et "linux_by_ssh_advanced" sur lesquels sont accrochés les différents checks dédiés contiennent des données (*locales*) qui seront utilisées par les checks. Ces données seront invoquées par les checks et commandes via **$_HOST** suivi du nom de la variable.

Exemple : **$_HOSTSSH_KEY$** utilisera la donnée nommée **SSH_KEY** (*qu'elles soient locales ou héritées d'un modèle*).

Pour un hôte qui hérite par exemple du modèle "linux_by_ssh" ou "linux_by_ssh_advanced" de notre pack, ces données seront donc héritées également, mais elles pourront aussi être surchargées directement sur l'hôte (*attention aux conflits de nom des données*).

Si vous souhaitez modifier de manière globale ces données, ou en rajouter, faites-le directement sur le modèle linux voulu, ceci s'appliquera alors à tous vos hôtes utilisant ce modèle.

Pour plus d'information, veuillez consulter la page sur les Les Variables (Remplacement dynamique de contenu - Anciennement les Macros).



Comment utiliser le pack Linux by SSH

Le pack linux_by_ssh peut être utilisé en appliquant le modèle linux souhaité à un hôte. Il existe deux manières de procéder :

En utilisant l'interface de Configuration

Dans l'interface de Configuration, créez ou éditez un host, et ajoutez le modèle "**linux_by_ssh"** ou **"linux_by_ssh_advanced"** dans la propriété "**Modèles d'hôte hérités**" à l'aide du menu déroulant.
(*voir la page* Éditer un Hôte)

En éditant les fichiers de configuration d'un collecteur (cfg)

Dans un fichier de configuration, créez ou éditez votre définition d'hôte en ajoutant, dans la propriété "**use",** la valeur "**linux_by_ssh"** ou **"linux_by_ssh_advanced"** selon les besoins.

Le fichier de configuration devra alors être importé avec une source (*plus d'information sur cette page:* Collecteur de type (cfg-file-import) - Import depuis des fichiers au format .cfg).

Configuration de la connexion SSH

Pour l'exécution correcte des commandes Linux, vous aurez besoin d'une connexion SSH.

Quelques informations au préalable sont nécessaires pour la bonne compréhension de cette partie.

D'une part, du côté de l'architecture Shinken, l'exécution des commandes est réalisée par les Pollers, en tant qu'utilisateurs "**shinken**". Comme pour tous les serveurs hébergeant Shinken, cet utilisateur est un utilisateur sans mot de passe par défaut. (les connexions SSH vers les serveurs Shinken via cet utilisateur ne sont donc possibles qu'avec une clé SSH)

D'autre part, du côté des machines Linux supervisées, un nom d'utilisateur, et une clé SSH ou mot de passe sont requis. Dans le modèle Linux, des données sont prévues à cet effet.

Nous conseillons l'utilisation d'un utilisateur spécifique (pour le service de supervision) ainsi que l'utilisation d'une connexion via clé SSH, afin d'éviter l'utilisation du super utilisateur root qui n'est pas requis par les checks.

Remarque

Si vous utilisez le pack linux pour superviser vos serveurs hébergeant Shinken, vous pouvez utiliser l'utilisateur déjà créé et utilisé par Shinken Entreprise : **shinken.**

Si vous choisissez cet utilisateur, vous n'aurez pas besoin de données particulières pour vos modèles d'hôte "linux_by_ssh" ou "linux_by_ssh_advanced", car les valeurs par défaut à l'installation de shinken suffiront (voir le tableau de données plus bas dans cette page).

Il faudra par contre réaliser les autorisations manuelles via clé SSH.

Cas Particulier

Par défaut, vos serveurs Shinken autorisent les connexions SSH émises par l'utilisateur **shinken** de leur propre serveur.

Donc dans le cas d'une installation rapide, le Poller pourra exécuter avec succès les requêtes SSH envoyées sur lui-même, sans que vous ne fassiez de manipulations avec les clés.

Côté client (machine ou serveur linux supervisé)

Si votre utilisateur de supervision n'est pas déjà créé sur votre linux à superviser, depuis un terminal de la machine supervisée "**linux-1**" (avec l'utilisateur root), il faut créer un nouvel utilisateur local avec mot de passe.

Dans cet exemple, il s'agit de "user-service-shinken" mais vous pouvez créer un autre utilisateur.

```
              [root@linux-1 ~]# adduser -m -r user-service-shinken
[FACULTATIF] : [root@linux-1 ~]# passwd user-service-shinken
```

Notez que la mise en place d'un mot de passe pour cet utilisateur n'est pas obligatoire, mais il vous faudra copier la clé SSH via la **méthode manuelle** expliquée plus bas, car la commande automatique ssh-copy-id requiert un mot de passe pour l'utilisateur du système de destination.

Côté serveur Poller

Copie de la clé SSH de votre utilisateur de supervision "**user-service-shinken**" depuis le serveur Poller "**shinken-poller**" (pour cet exemple), vers le serveur supervisé "**linux-1**" (*dans cet exemple, ip 192.168.1.19*)

Copie clé SSH via commande ssh-copy-id

Soit via la méthode "automatique" via la commande ssh-copy-id en se connectant au préalable via l'utilisateur **shinken** sur le ou les serveurs pollers :

```
[root@shinken-poller ~]# su - shinken
[shinken@shinken-poller ~]#  ssh-copy-id -i ~/.ssh/id_rsa user-service-shinken@linux-1
The authenticity of host '192.168.1.19 (192.168.1.19)' can't be established.
RSA key fingerprint is 00:ff:ee:dd:cc:bb:aa:d6:d3:79:1d:f6:93:47:80:27.
Are you sure you want to continue connecting (yes/no)? yes
user-service-shinken@linux-1's password: XXXXXXXXXX
Now try logging into the machine, with "ssh 'user-service-shinken@linux-1'", and check in:
  .ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.
```

Copie clé SSH via commande ssh

Soit via une commande SSH depuis le serveur Poller, il s'agit d'ajouter la clé publique au fichier "authorized_keys" du serveur supervisé (ici vm2) :

```
cat /var/lib/shinken/.ssh/id_rsa.pub | ssh root@vm2 "cat >> /var/lib/shinken/.ssh/authorized_keys"
```

Ici la connexion se fait via l'utilisateur root du serveur vm2 (mais vous pouvez utiliser votre propre utilisateur), le but étant de rajouter, en une commande SSH, la clé de l'utilisateur shinken du Poller **/var/lib/shinken/.ssh/id_rsa.pub** à la fin du fichier **/var/lib/shinken/.ssh/authorized_keys** du serveur supervisé.

Copie clé SSH manuellement

Soit via méthode "manuelle" via rajout de la clé dans le fichier authorized_keys

- Récupérez la clé publique de l'utilisateur qui va établir la connexion SSH, et la copier

```
[root@shinken-poller ~]# su - shinken
[-bash-4.1]$ less .ssh/id_rsa.pub

-> copiez la clé
```

- Connectez vous sur le serveur linux supervisé avec votre utilisateur de supervision et collez cette clé dans le fichier "authorized_keys" de l'utilisateur de supervision:

```
[root@linux-1 ~]# su - user-service-shinken
[-bash-4.1]$ vi .ssh/authorized_keys

-> collez la clé
```

Test de connexion

Test de connexion au serveur "**remote-host**" en tant qu'utilisateur user-service-shinken via l'utilisateur du Poller (shinken) :

```
[root@shinken-poller ~]# su - shinken
[shinken@shinken-poller ~]# ssh user-service-shinken@linux-1 -i .ssh/id_rsa
```

La connexion doit s'établir avec succès.

Côté interface de configuration

Dans chaque hôte héritant du modèle d'hôte "linux_by_ssh" ou "linux_by_ssh_advanced", vous aurez 4 données concernant la connexion SSH, ces 4 données seront par la suite utilisées par tous les checks "linux_by_ssh" et "linux_by_ssh_advanced". Contrairement aux autres données, les valeurs par défaut de celles-ci sont configurées dans un certain fichier en central (serveur hébergeant l'Arbiter) **/etc/shinken/resource.d/ssh.cfg**.

Donnée	Description	Valeur par défaut	Valeur par défaut à l'installation de shinken
SSH_KEY	Répertoire de la clé générée sur votre serveur hébergeant le démon Poller	SSH_KEY	~/.ssh/id_rsa
SSH_KEY_PASSPHRASE	Mot de passe utilisé pour l'authentification de l'utilisateur ou pour utiliser la clé privée ("Passphrase") si nécessaire	$SSH_KEY_PASSPHRASE$	''
SSH_PORT	Port de connexion SSH	SSH_PORT	22

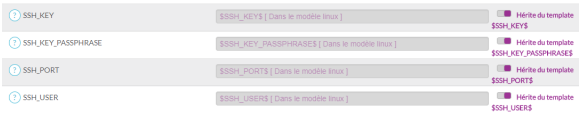
SSH_USER	Utilisateur pour la connexion SSH	SSH_USER	shinken

Remarque

Toutes les valeurs par défaut renvoient à une globale (cf Les Variables (Remplacement dynamique de contenu - Anciennement les Macros)) qui sont modifiables dans le fichier **/etc/shinken/resource.d/ssh.cfg**, attention cependant, la modification dans le fichier direct entraînera une modification sur tous les hôtes utilisant ces globales.

La modification des valeurs par défaut présentes dans le fichier du serveur (/etc/shinken/resource.d/ssh.cfg) nécessite un redémarrage intégrale du service shinken (service shinken restart).

Par exemple, voici le paramétrage d'une connexion via clé SSH par défaut :



Nom	Valeur étendue
_HOSTSSH_KEY	~/.ssh/id_rsa
_HOSTSSH_KEY_PASSPHRASE	''
_HOSTSSH_USER	shinken

Par exemple, voici le paramétrage d'une connexion via Utilisateur/Mot de passe :

Pour information, détail sur le chiffrement de la connexion

La librairie utilisée pour se connecter en SSH est la librairie paramiko (https://pypi.org/project/paramiko/)

La version utilisée est la:

- 1.18.5 (de 2018) à partir de la version v02.08.02
  - les versions supérieures ne sont pas compatibles avec la RH6
  -
    - aes128-ctr
    - aes192-ctr
    - aes256-ctr
    - aes128-cbc
    - blowfish-cbc
    - aes192-cbc
    - aes256-cbc
    - 3des-cbc
    - arcfour128
    - arcfour256
  -
    - diffie-hellman-group1-sha1
    - diffie-hellman-group14-sha1
    - diffie-hellman-group-exchange-sha1
    - diffie-hellman-group-exchange-sha256
- 1.15.1 (de 2014) avant la v02.08.02
  -
    - aes128-ctr
    - aes256-ctr
    - aes128-cbc
    - blowfish-cbc
    - aes256-cbc
    - 3des-cbc
    - arcfour128
    - arcfour256
  -
    - diffie-hellman-group14-sha1
    - diffie-hellman-group-exchange-sha1
    - diffie-hellman-group1-sha1

Commandes additionnelles

Dans l'idée d'apporter un service personnalisé, certaines commandes n'ont volontairement pas été ajoutées en tant que check dans le pack linux_by_ssh. En effet des scripts permettent au cas par cas de rajouter des options spécifiques à votre supervision et ne sont donc pas compatibles avec tous les types de machines. Pour éviter donc de vous proposer un service qui ne vous sera peut-être pas utile, nous avons mis à votre disposition la commande simple que vous pourrez utiliser en créant un check.
Pour plus d'informations, rendez-vous sur la page Commandes additionnelles

Version des scripts livrés

Nom du script	Version
check_cpu_stats_by_ssh.py	0.1
check_disks_by_ssh.py	0.1
check_disks_stats_by_ssh.py	0.1
check_kernel_stats_by_ssh.py	0.1
check_load_average_by_ssh.py	0.1
check_memory_by_ssh.py	0.1
check_net_stats_by_ssh.py	0.1
check_nfs_stats_by_ssh.py	0.1
check_ntp_sync_by_ssh.py	0.1
check_ro_filesystem_by_ssh.py	0.1
check_uptime_by_ssh.py	0.1
check_ssh_connexion.py	0.1
check_tcp_states_by_ssh.py	0.1