Corrélation des problèmes sources et impacts

Sommaire

Qu'est ce que la corrélation?

L'objectif de la corrélation est de proposer des aides à l'utilisateur afin qu'il arrive facilement à faire la distinction entre:

- les problèmes sources, qui sont des éléments qui sont en erreur, de leur propre fait, par exemple
  - une application qui est tombée
  - un serveur qui est arrêté
- les impacts: ce sont des éléments qui sont en erreurs, mais du fait d'un ou plusieurs problèmes sources, par exemple:
  - une application sur un serveur qui s'est arrêté
  - un serveur démarré, mais derrière des switchs réseaux qui sont tombés

Le principe est que pour revenir à une situation saine, il faut régler les problèmes sources, car redémarrer les impacts n'aura aucun effet tant que l'on n'aura pas réglé son/ses problème(s) sources.

Application du principe de corrélation

Notifications

Les notifications appliquent ce principe en n'envoyant qu'une notification dans le cas d'un problème source.

Prenons par exemple, un service web présent sur un serveur est lui connecté via un switch.

- Si le switch tombe, le serveur et donc le service web ne sont plus disponibles.
- Au lieu d'envoyer une notification pour chaque partie de l'infrastructure (serveur, service web et switch), la corrélation permet de n'envoyer qu'une notification pour l'élément à l'origine du problème c'est-a-dire le switch.

Les checks deviennent INCONNU si leur parent est CRITIQUE (Optionnel)

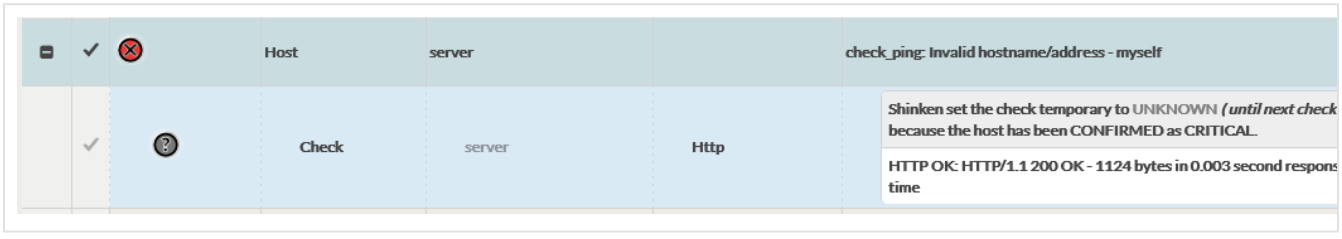
Les vérifications des checks et des hôtes n'étant pas fait en même temps afin de ne pas surcharger l'hôte,
on peux avoir le cas suivant où

- un check (par exemple CPU) retourne un OK,
- la vérification de l'hôte arrive en CRITIQUE, et l'hôte est certifié tombé (on a atteint son nombre maximum de tentative de vérification)
- on a donc sur l'interface:
  - un hôte en CRITIQUE,
  - un check qui est OK,
  - ceci est très incohérent visuellement

Pour corriger cette incohérence qui va durer jusqu'à la prochaine vérification du check, le démon Scheduler est capable de changer temporairement l'état des checks de l'hôte:

- il les mets en INCONNU, car vu ce qu'il a détecté sur l'hôte, il y a un doute raisonnable concernant les états des checks
- mets un texte explicatif au début du résultat du check concernant cette modification

Visuellement, le check apparaitra de la manière suivante, indiquant que son état a été modifié par Shinken:



Dès la prochaine vérification du check, ce dernier prendra son état et résultat définitif (qui peux être en erreur ou pas)

Il est important de savoir que cette approche ne remet pas en cause la logique de gestion HARD/SOFT : il n'est pas pris en compte dans la gestion des tentatives de checks.

Par contre dans un souci de cohérence, cette état / résultat sera également disponible dans les autres modules, genre SLA ou les Événements.

Comment activer / désactiver ce mécanisme

Par défaut ce mécanisme est activé. On peux le désactiver dans le fichier */etc/shinken-user/configuration/daemons/arbiters/arbiter_cfg_overload.cf*g (qui surcharge le fichier */etc/shinken.shinken.cfg*) en mettant le paramètre:

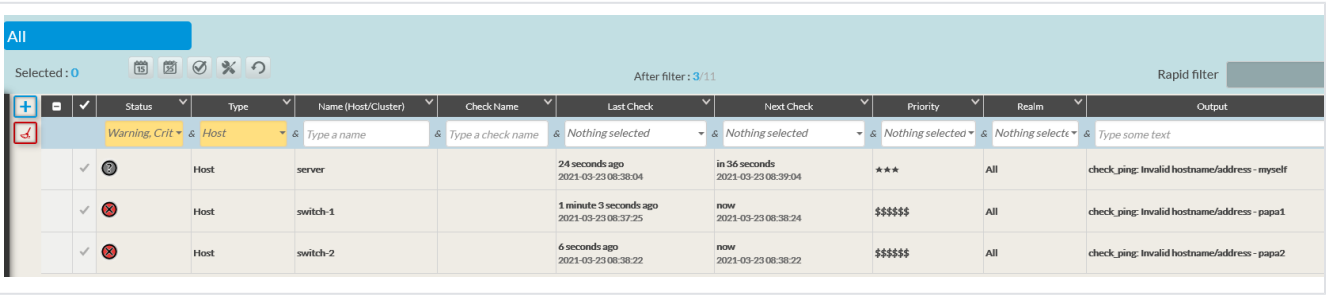
```
enable_problem_impacts_states_change=0
```

Tous les parents d'un hôte sont tombés

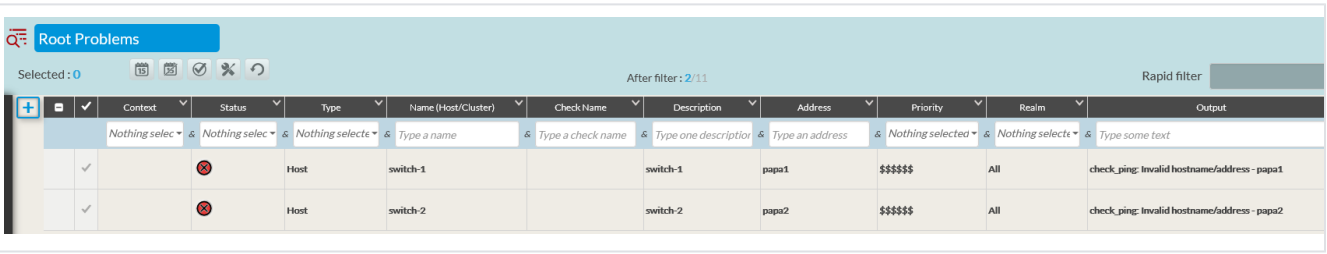
Dans le cas où on a des dépendances réseaux (entre les hôtes donc), la règle d'assignation des problèmes sources / impacts est la suivante:

- si un hôte a au moins un parent, et que tous ses parents sont en erreur, alors il sera affiché comme INCONNU :
  - car en fait il n'y a aucun chemin réseau pour le contacter, donc on ne sait pas dans quel état il est réellement.
  - pour les gestionnaires d'événements, Nagvis et le module livedata, cet état est nommé INJOINGNABLE (UNREACHABLE).
- s'il n'a pas de parents, ou qu'au moins un de ses parents est disponible, alors il sera affiché comme CRITIQUE,
  - car il y a bien un chemin réseau pour lui parler, et donc s'il ne répond pas c'est que c'est bien sa faute

Dans le premier cas, on aura l'affichage suivant dans la liste complète, l'hôte "server" ayant pour parents switch-1 et switch-2.



Si on passe par la liste qui préfiltre les problèmes sources, on aura alors que les deux hôtes qui sont les problèmes sources:



Cette vue peut donc être très pratique quand les relations de dépendances sont configurées.

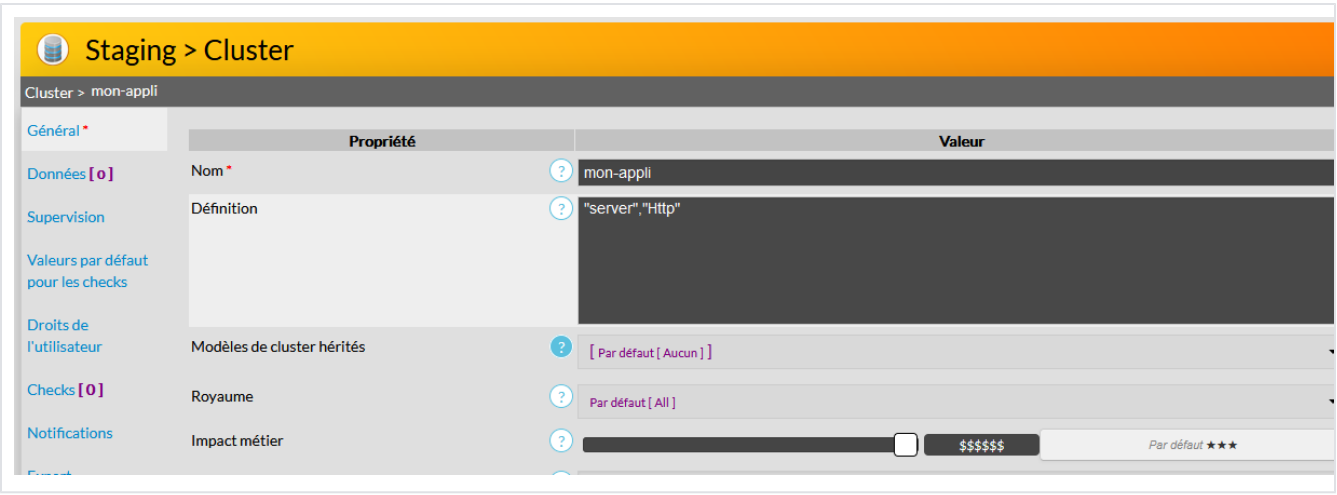
Impact métier dynamique

L'impact métier permet de déterminer si un problème source est plus ou moins important, en sachant s'il a cassé quelque chose d'important.

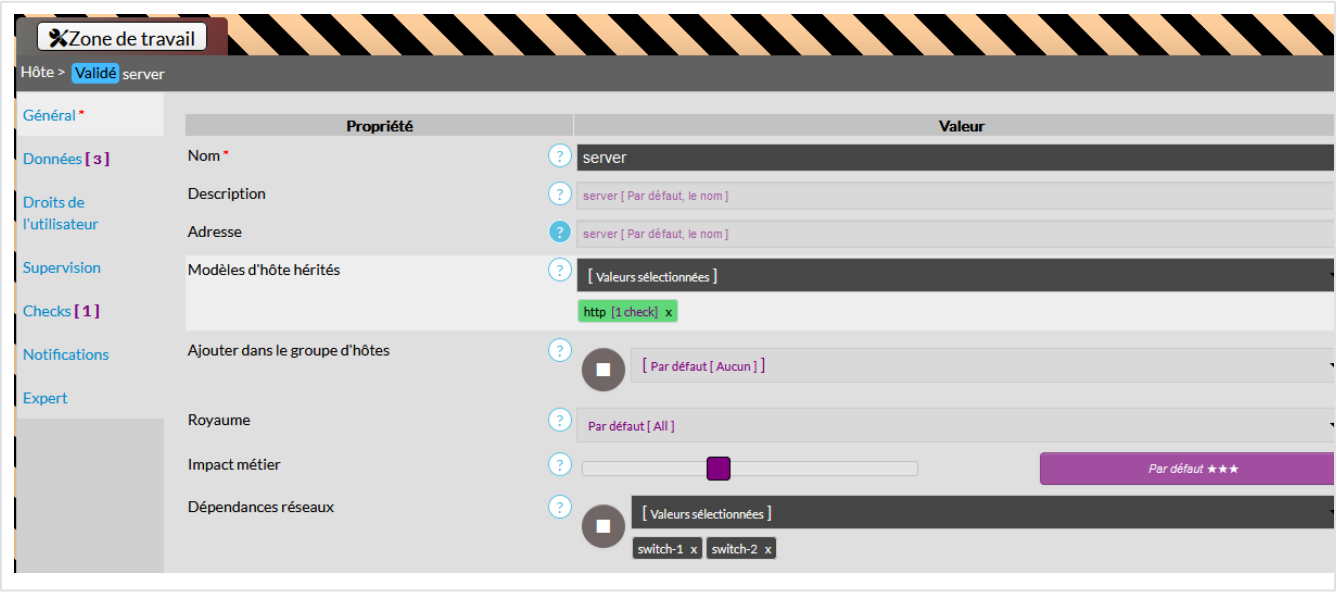
Si les liens de dépendances (clusters, parents) sont définis dans Shinken, alors ce dernier va être capable de propager les impact métier, des impacts vers leurs problèmes sources.

Dans notre exemple, nous avons un cluster (mon-appli) qui repose sur un simple serveur Http, dont l'hôte (server) dépend de deux swtichs (switch-1 et switch-2).

Dans la configuration, seul le cluster a un impact métier important:



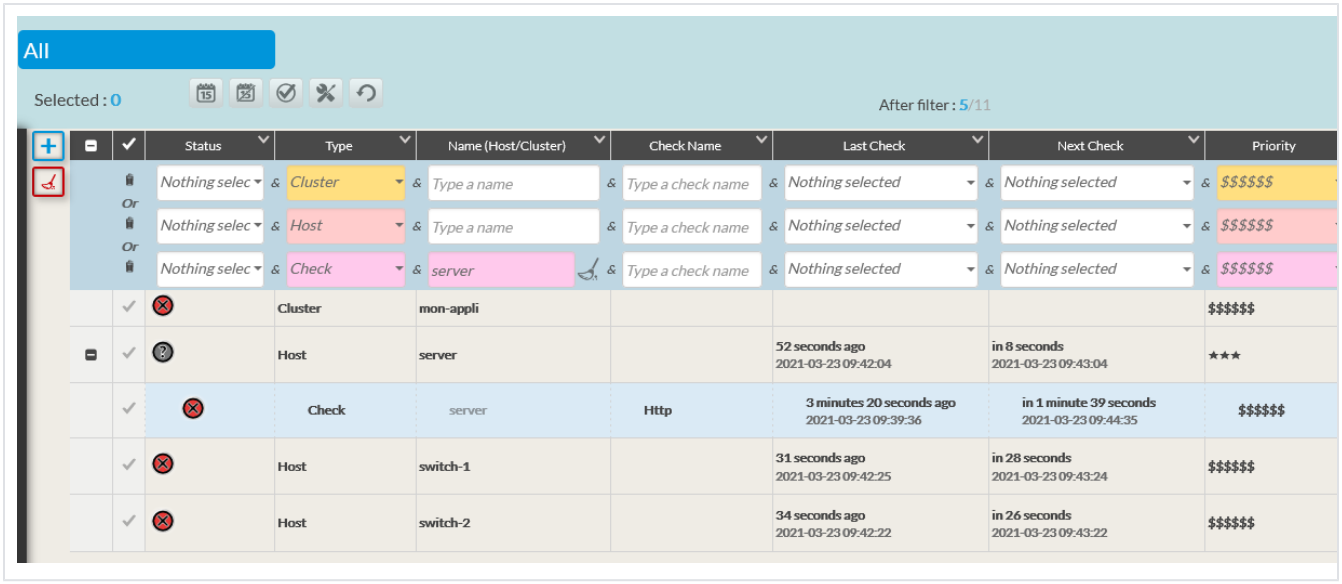
Les hôtes server, switch-1 et switch-2 ont quant à eu des impacts métiers normaux, par exemple notre hôte server:



Quand on regarde du côté de l'interface de visualisation, on remarque que Shinken a mis à jour l'ensemble des impacts métiers de la chaine, de proche en proche:

- Le check Http sur l'hôte server ayant cassé le cluster avec un haut impact métier, il gagne lui-même cette valeur
- L'hôte server ayant cassé son check, il gagnera lui-même une haute valeur
- Les switch-1 et switch-2 ont étés détectés comme ayant cassé l'hôte server, ils héritent également de la haute importance métier.

Ceci se traduit dans l'interface de visualisation par une colonne Priorité qui a sa valeur au maximum, comme celle du cluster mon-appli:


All
Selected : 0
After filter : 5/11
Status
Type
Name (Host/Cluster)
Check Name
Last Check
Next Check
Priority
Nothing selec
Cluster
Type a name
Type a check name
Nothing selected
$$$$$$
Or
Host
Check
server
mon-appli
$$$$$
Host
server
52 seconds ago
2021-03-23 09:42:04
in 8 seconds
2021-03-23 09:43:04
★★★
Check
server
Http
3 minutes 20 seconds ago
2021-03-23 09:39:36
in 1 minute 39 seconds
2021-03-23 09:44:35
$$$$$$
Host
switch-1
31 seconds ago
2021-03-23 09:42:25
in 28 seconds
2021-03-23 09:43:24
$$$$$$
Host
switch-2
34 seconds ago
2021-03-23 09:42:22
in 26 seconds
2021-03-23 09:43:22
$$$$$$