Haute disponibilité des démons de Shinken Entreprise

Sommaire

- Haute disponibilité pour la supervision
- Comment configurer Shinken Entreprise pour avoir une supervision hautement disponible
  - Le principe des Spares
  - Le rôle central de l'Arbiter
  - Procédure de configuration d'un démon Spare
  - Cas particulier du Broker
  - Cas particulier de l'Arbiter/Synchronizer
    - Fonctionnement de l'Arbiter Spare
    - Mise en place d'un Arbiter Spare
- Visualiser l'état d'un spare
  - Shinken-healthcheck
  - Checks Shinken
- Conseils d'utilisation
  - Paramétrage des délais de vérification
  - Pollers spares

Haute disponibilité pour la supervision

La plateforme de supervision est souvent un élément très important d'un parc informatique, vous permettant de rester proactif sur votre infrastructure. Il faut pouvoir avoir pleinement confiance en cette plateforme, en sa capacité à informer des erreurs et problèmes potentiels du périmètre des machines supervisées.

Il est donc intéressant d'avoir une architecture hautement disponible pour la supervision afin que celle-ci soit fonctionnelle même lors d'un problème sur une partie de votre réseau.

Shinken Entreprise permet de configurer les démons qui composent une installation de manière à obtenir une supervision plus fiable et résistante aux pannes.

Comment configurer Shinken Entreprise pour avoir une supervision hautement disponible

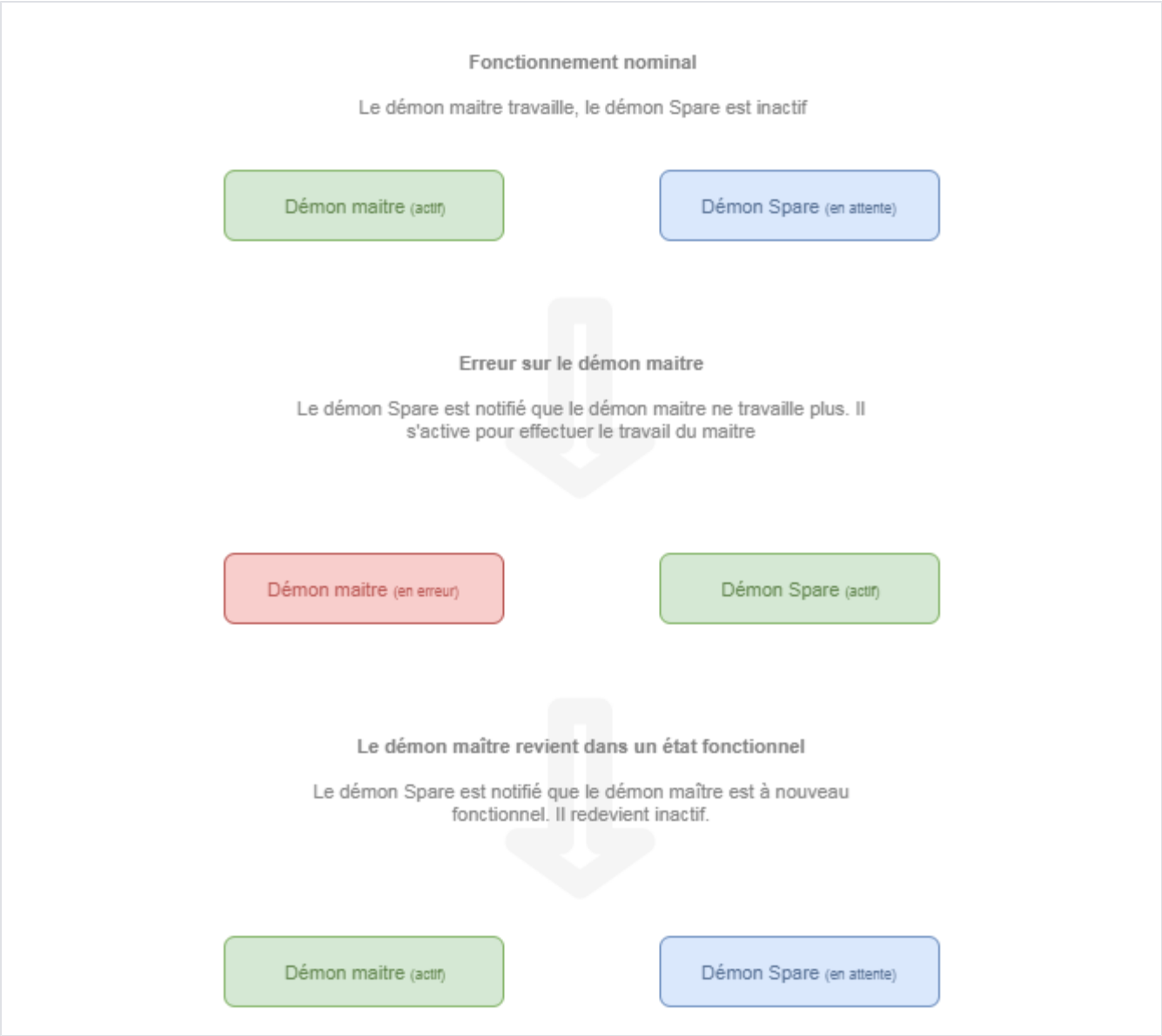
Le principe des Spares

Chaque démon dans Shinken Entreprise peut être secondé par un démon de même type configuré en tant que Spare. Le principe de Spares permet d'assurer une haute disponibilité en assurant un service continu même lorsqu'un démon cesse de fonctionner.

On classe alors les démons dans 2 catégories: les démons maîtres et les démons Spare.

Le cycle de vie d'un Spare est le suivant :

- Dans un fonctionnement nominal, le démon maître fonctionne correctement: il reçoit des données des autres démons et effectue son travail. Le démon Spare est présent pour pouvoir effectuer le travail du démon maître si celui-ci ne peut plus assumer son rôle. Dans un cas de fonctionnement nominal comme celui-ci, le démon Spare est inactif.
- Le démon maître cesse de fonctionner pour une raison quelconque (*erreur inconnue, arrêt volontaire, maintenance du serveur ...*). Il ne peut donc plus assumer son rôle. Ce dysfonctionnement est détecté par l'Arbiter qui va alors activer le démon Spare pour qu'il remplace le démon maître et puisse continuer d'assurer le service. Le démon Spare agit donc exactement comme le démon maitre: il assure le même service. Ce remplacement de démons est invisible du point de vue de l'utilisateur.
- Le démon maître est à nouveau fonctionnel (*intervention suite à une erreur, fin de maintenance du serveur ...*). L'Arbiter remarque que le démon maître peut à nouveau effectuer son travail. Puisqu'il s'agit du démon maître, il va être préféré au démon Spare. L'Arbiter va donc désactiver le démon Spare et ordonner au démon maître de reprendre son travail.



Plusieurs remarques peuvent être effectuées sur le fonctionnement des Spares.

- L'exemple sur le schéma met en scène un seul démon maître et un seul démon Spare. Il peut en fait exister **plusieurs démons maîtres et plusieurs démons Spares.**
  Par exemple, on peut avoir 3 Schedulers maîtres et 2 Schedulers spares. Dans ce cas, lorsqu'un Scheduler maitre est en erreur, un Scheduler spare sera choisi au hasard pour assurer le rôle du Scheduler maître en erreur.
  - Un cas à part est le démon Broker, pour qui, il faut désigner dans sa configuration quel Broker spare prendra son relai. Plus de détail plus bas dans la page.
- Un démon Spare ne peut remplacer un démon maître que si ils sont **de même type**.
  Par exemple, un Scheduler Spare ne peut pas remplacer un Poller maître en erreur, mais il pourra remplacer un Scheduler maître.
- Un démon Spare ne peut remplacer un démon maître que si les deux démons se trouvent **dans le même royaume ou sous-royaume**.
  Aussi, un démon Spare peut remplacer un démon d'un sous-royaume seulement si l'option "*manage_sub_realm*" est activée dans le fichier de configuration du démon. Un Poller Spare d'un royaume "France" ne pourra pas remplacer un Poller maître d'un royaume "Allemagne".

Le rôle central de l'Arbiter

Le premier rôle de l'Arbiter est de compiler la configuration et de l'envoyer aux différents démons qui composent l'installation Shinken. Comme son nom l'indique, il agit en tant qu'arbitre central et propage la configuration aux démons.

En tant que démon central dans l'architecture Shinken, il vérifie quel démon fonctionne et ne fonctionne pas, et déclenche l'activation d'un démon Spare si besoin.

Le fonctionnement de la haute disponibilité des démons est donc complètement lié et dépendant de l'Arbiter : les démons Spare ne s'activent pas tout seuls, mais sur ordre de l'Arbiter qui leur ordonne de prendre le relais d'un démon qui ne fonctionne pas.

Sans démon Arbiter en fonctionnement dans l'architecture, le mécanisme de haute disponibilité des démons ne peut pas fonctionner.

La mise en place d'un Arbiter Spare est donc conseillée pour assurer une haute disponibilité dans une architecture Shinken.

Procédure de configuration d'un démon Spare

L'ajout d'un démon Spare s'effectue en 4 étapes:

- Déclarer le démon comme n'importe quel autre démon.
- Activer la propriété Spare dans le démon.
- S'assurer que le démon en question est bien activé sur la machine du démon Spare.
- Redémarrer Shinken afin que la nouvelle configuration soit prise en compte par les démons.

Pour illustrer la configuration d'un démon Spare, on prend l'exemple d'un Scheduler qu'on va configurer en tant que Spare.

La configuration d'un Spare, comme pour la configuration d'un démon classique, s'effectue sur la machine hébergeant le démon Arbiter. Aucune modification au niveau des fichiers de configuration n'est à effectuer sur la machine hébergeant le Scheduler Spare.

- **Déclaration du Scheduler**
  La première étape est donc de déclarer le nouveau Scheduler dans **/etc/shinken/schedulers/scheduler-spare.cfg**. Dans ce fichier, on va donc mettre le contenu suivant (*copié depuis* ***/etc/shinken/schedulers/scheduler-master.cfg***).

/etc/shinken/schedulers/scheduler-spare.cfg

```
define scheduler {
    scheduler_name               scheduler-spare
    address                            adresse-scheduler-spare (ip ou nom dns)
    port                               7768
    use_ssl                            0

    timeout                            3
    data_timeout                       120
    max_check_attempts                 3
    check_interval                     60

    modules                            MongodbRetention

    realm                              All
    spare                              0

    enabled                            1
}
```

- **Mise en place du paramètre spare**
  On prend soin ensuite de passer la valeur de l'option **spare** à 1

/etc/shinken/schedulers/scheduler-spare.cfg

```
define scheduler{
    ...
    spare 1
    ...
}
```

- **Activation du Scheduler sur la machine distante**

Enfin, sur la machine hébergeant le Scheduler Spare, il faut vérifier que le Scheduler est bien activé. Cette vérification va se faire avec les commandes de manipulation des démons (*voir la page :* Manipulation des démons Shinken).
On vérifie d'abord que le démon Scheduler est activé (*sur la machine du Scheduler*) :

```
$ shinken-daemons-list
```

Si le Scheduler n'est pas activé sur la machine Spare, on peut l'activer grâce à la commande *shinken-daemons-enable* :

```
$ shinken-daemons-enable scheduler
```

Aussi, seuls les démons utilisés doivent être activés sur la machine de Spare, sous peine d'avoir des comportements hasardeux en cas de mauvaise configuration. Les démons inutilisés peuvent être désactivés grâce à la commande *shinken-daemons-disable* :

```
$ shinken-daemons-disable demon1 demon2 ...
```

Cas particulier du Broker

Dans le cas du Broker, vu qu'il gère un ensemble de modules bien identifiés, il faut que l'administrateur définisse plus finement quel démon prendra la main lorsque le démon master sera déclaré tombé.

- Le démon Spare pourra avoir ses propres modules (*avec des adresses spécifiques : vu que le positionnement réseau peut être différent*).
  - Mais ces modules doivent être du même type que ceux du démon master qu'il remplace.
  - Par exemple: WebUI et Sla sur le démon master, WebUI-spare et Sla-spare sur le démon spare.
  - Cette restriction peux être supprimée si on mets le paramètre **broker__manage_spare__spare_must_have_the_same_list_of_module_type** à 0 dans le .cfg du master,
    - ceci permet d'avoir un spare potentiellement en mode dégradé.
- Un démon Spare ne peut être désigné que par un et un seul démon master.

La désignation d'un Spare pour un Broker se fait à l'aide de la propriété **spare_daemon** qui prend le nom du démon spare.

- **Déclaration du broker**
  La première étape est donc de déclarer le nouveau Broker dans **/etc/shinken/brokers/broker-spare.cfg**. Dans ce fichier, on va donc mettre le contenu suivant (*copié depuis* ***/etc/shinken/brokers/broker-master.cfg***).

/etc/shinken/brokers/broker-spare.cfg

```
define broker {
    broker_name                 broker-spare
    address                           adresse-broker-spare (ip ou nom dns)
    port                              7772
    use_ssl                           0

    ...

    modules                           WebUI,sla

    realm                             All
    spare                             0

    enabled                           1
}
```

- **Mise en place du paramètre spare**
  On prend soin ensuite de passer la valeur de l'option **spare** à 1

/etc/shinken/brokers/broker-spare.cfg

```
define broker{
    ...
    spare 1
    ...
}
```

- **Vérification des modules**
  Si le broker Spare est situé dans un autre réseaux que le démon master, peux être que les adresses de connexions vont aussi être différente *(par exemple à la base Mongo*), si besoin, il faut donc copier les modules qui en ont besoin, et mettre dans le broker-spare ces nouveaux modules :

/etc/shinken/brokers/broker-spare.cfg

```
define broker{
    ...
    modules                             WebUI,sla-spare
    ...
}
```

- **Désignation dans le broker-master de son spare**
  Il suffit alors de mettre dans le broker-master le nom du Spare que nous venons de créer pour qu'ils soient liés, ceci via la propriété **spare_daemon** :

/etc/shinken/brokers/broker-spare.cfg

```
define broker{
    broker_name             broker-master
    ....
    spare_daemon            broker-spare
    ...
}
```

Cas particulier de l'Arbiter/Synchronizer

Nous avons vu que dans Shinken Entreprise, l'ajout d'un démon spare n'était pas plus difficile que l'ajout de n'importe quel autre démon :

- Déclaration du démon.
- Activation du paramètre Spare.
- Activation du démon sur la machine concernée.

Deux démons échappent à cette règle de configuration pour le cas des Spares : l'Arbiter et le Synchronizer.

En effet, ces deux démons sont spéciaux dans le sens ou ils ne peuvent pas être répliqués facilement :

- Le Synchronizer est unique dans une installation Shinken Entreprise. Il permet de définir la configuration et contient les informations sur les éléments supervisés dans une base de données locales. Le dupliquer ou l'utiliser en tant que Spare est bien plus complexe que pour les autres démons, car il faut également répliquer la base de données de configuration.
- L'Arbiter est le démon central de Shinken Entreprise. Il distribue la configuration à tous les autres démons et nécessite des étapes de configuration supplémentaires pour être utilisé en tant que Spare.

Important

- Il ne peut y avoir qu'un seul Arbiter master et un seul Arbiter spare par infrastructure.
- Sur l'Arbiter spare, le module architecture-export ne générera aucune architecture (voir la page : Visualiser l'architecture de son installation Shinken)

Pas de Synchronizer Spare

Compte tenu des spécificités du rôle du Synchronizer, de ses modules et de sa communication particulière avec le démon Arbiter, **le mécanisme de Spare n'est actuellement pas géré dans Shinken Entreprise pour le démon Synchronizer**.

Fonctionnement de l'Arbiter Spare

Comme mentionné précédemment, le rôle central de l'Arbiter nécessite des adaptations lorsqu'on veut mettre en place le mécanisme de Spare.

Dans une architecture standard, l'Arbiter surveille les autres démons. Dès que l'un d'eux ne répond plus, il déclenche la bascule vers un démon Spare et envoie la configuration au démon Spare qui va prendre le relai.

Dans le cas d'un Arbiter Spare, ce mécanisme n'est pas applicable puisque dans le cas où l'Arbiter Master entre en erreur, il ne peut pas donner à l'Arbiter Spare sa configuration et l'ordre d'activation.

Le fonctionnement d'un Arbiter Spare est donc différent du mécanisme de haute disponibilité des autres démons.

- L'Arbiter Master envoie chaque changement de configuration à l'Arbiter Spare. L'Arbiter Spare prend cette configuration en mémoire et l'enregistre également dans une rétention pour l'avoir localement en cas de redémarrage.
- L'Arbiter Spare, comme les autres démons Spare sont inactifs. Il reçoit à intervalle régulier une communication de l'Arbiter Master qui lui indique qu'il est en bon état de fonctionnement.
- Lorsque l'Arbiter Master ne fonctionne pas, l'Arbiter Spare ne reçoit plus de communication régulière de la part de l'Arbiter Master et considère que celui-ci ne fonctionne plus. Il prend alors le rôle de l'Arbiter Master en utilisant la configuration précédemment reçue.

Ce fonctionnement a les conséquences suivantes :

- Le fonctionnement d'un Arbiter Spare ne nécessite pas d'accès au Synchronizer ni à la base Mongo. Puisque la configuration est chargée en mémoire ou bien disponible dans une rétention, elle est disponible rapidement sans avoir besoin de contacter le Synchronizer et la base de données de configuration.
- Quand l'Arbiter Master n'est plus disponible, l'Arbiter Spare a pour rôle de maintenir la plateforme de supervision en l'état actuelle (*gestion des spares et de la configuration en cas de redémarrage d'un démon*). Il s'agit d'un mode de fonctionnement dégradé ce qui rend l'application d'une nouvelle configuration en production depuis l'interface de Configuration impossible. Le retour en fonctionnement de l'Arbiter Master permet à nouveau l'application d'une configuration via l'interface de Configuration.

Mise en place d'un Arbiter Spare

Compte tenu de son rôle central, la configuration d'un Arbiter Spare nécessite donc des étapes supplémentaires :

- Déclaration d'un nouvel Arbiter dans la configuration.
- Activation du paramètre Spare.
- Mise en place du paramètre host_name dans la configuration des 2 Arbiters.
- Activation du démon sur la machine concernée.
- Activation du paramètre Spare sur la machine hébergeant l'Arbiter Spare.
- Création d'une sauvegarde de la configuration.

- **Déclaration du nouvel Arbiter dans la configuration**

/etc/shinken/arbiters/arbiter-spare.cfg

```
define arbiter {
    arbiter_name         arbiter-spare

    host_name            hostname_machine_spare
    address              adresse-arbiter-spare

    port                 7770
    use_ssl              0

    timeout              3
    data_timeout         120
    max_check_attempts   3
    check_interval       60

    spare                0

    modules              synchronizer-import

    enabled              1
}
```

- **Activation du paramètre Spare**
  Dans le fichier de configuration de l'Arbiter Spare, on passe ensuite le paramètre Spare à 1.

/etc/shinken/arbiters/arbiter-spare.cfg

```
define arbiter {
    ...
    spare                1
    ...
}
```

- **Mise en place du paramètre host_name sur les 2 Arbiters**
  Chaque Arbiter dans la configuration doit pouvoir s'identifier. Puisqu'il ne peut y avoir qu'un seul Arbiter par machine, on choisit d'identifier chaque Arbiter par le hostname de la machine sur laquelle il est installé.
  Pour cela, on spécifie le nom du hostname de la machine dans le paramètre host_name de chaque Arbiter.
  Par exemple, sur la machine de l'Arbiter spare, on cherche le hostname de la machine :

```
$ hostname
hostname_machine_spare
```

Sur la machine hébergeant l'Arbiter master, donc la machine centrale sur laquelle s'effectue la configuration des démons, on spécifie pour chaque Arbiter le paramètre **host_name** (à effectuer pour les 2 Arbiters).

/etc/shinken/arbiters/arbiter-spare.cfg

```
define arbiter {
    ...
    host_name    hostname_machine_spare
    ...
}
```

- **Activation du démon sur la machine concernée**
  Comme pour les autres démons, il faut s'assurer sur la machine hébergeant le démon que celui-ci est bien activé. Cette opération se fait avec les commandes de manipulsation des démons (*voir la page :* Manipulation des démons Shinken) : *shinken-daemons-enable*, *shinken-daemons-disable* et *shinken-daemons-list*.

- **Activation du paramètre spare sur la machine hébergeant l'Arbiter spare**
  En règle générale, la configuration de tous les démons se fait sur la machine centrale de l'installation Shinken (*celle qui héberge l'Arbiter maître*).
  Le cas de l'Arbiter Spare est un cas particulier. Puisqu'il a un rôle central, lorsqu'il démarre, il agit automatiquement comme arbitre avec les autres démons. Dans le cas d'un spare, on veut par contre qu'il reste inactif et n'essaye pas d'envoyer de configuration aux autres démons. Pour cela, il faut, sur la machine du Spare, spécifier que le démon est en spare.

  Machine spare > /etc/shinken/arbiters/arbiter-master.cfg

  ```
  define arbiter {
      arbiter_name    arbiter-spare
      ...
      spare           1
      ...
  }
  ```

- **Création d'une sauvegarde de la configuration**
  Dans la crontab de votre serveur, il faut créer ajouter cette ligne :

  ```
  00 23 * * * root /usr/sbin/shinken-backup --configuration --output-directory /tmp/shinken-backup
  ```

  La sauvegarde de la configuration ne prend pas beaucoup de place, donc vous pouvez garder plusieurs jours.

  > En cas de crash du serveur il est préférable de conserver ces sauvegardes sur un autre emplacement que la machine hébergeant l'Arbiter Master (*idéalement directement sur l'Arbiter Spare*).

Visualiser l'état d'un spare

Le fait qu'un démon soit en spare est une information importante lorsqu'on veut déterminer le bon fonctionnement d'une plateforme de supervision Shinken Entreprise. La propriété spare d'un démon peut être vue à 2 endroits :

- lors d'une vérification du fonctionnement de l'installation avec le Shinken-healtcheck (*voir la page :* Shinken-healthcheck - Vérifier le bon fonctionnement de Shinken Entreprise),
- lors de la supervision de Shinken avec les checks du pack Shinken.

Shinken-healthcheck

Dans le Shinken-healthcheck (*voir la page :* Shinken-healthcheck - Vérifier le bon fonctionnement de Shinken Entreprise), les démons spare sont accompagnés d'une annotation "SPARE", qui permet de déterminer facilement si un démon est un démon maître un ou démon spare.

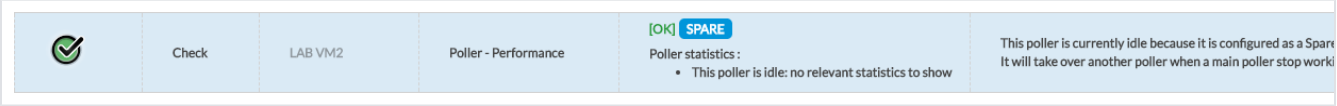
```
- 172.16.0.2 (172.16.0.2):
^^^^^^^^^^^^^^^^^^^^^^^^^^^
    [broker: broker-corse-spare] (SPARE)
        OK:      Configuration seems valid
        OK:      Connection to daemon is OK at port 7772
        OK:      Daemon version is: 02.04.01-003_BUILD07.fr
        OK:      Correct connection from arbiter "arbiter-vm3" ( and no time shift )
        Talk to:
```

Lors d'une prise de relais, une annotation "RUNNING" est rajoutée à la mention "SPARE".

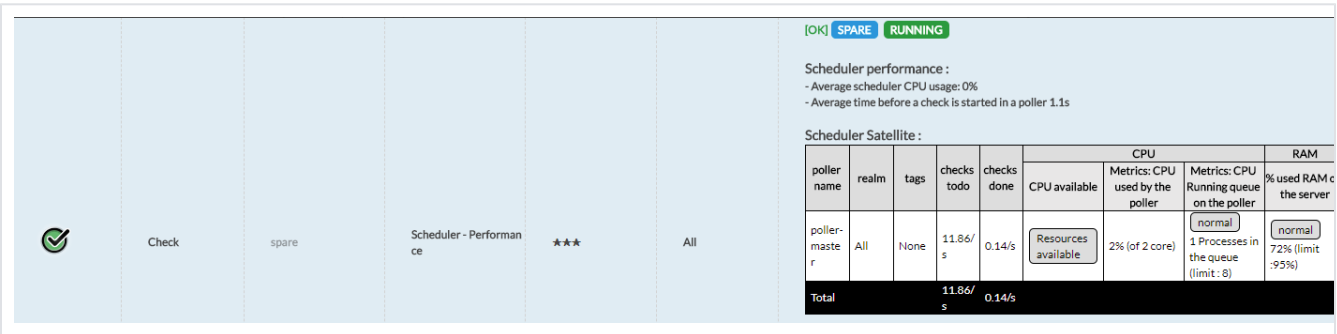
```
- 192.168.1.139 (192.168.1.139):
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    [scheduler: scheduler-spare] (SPARE) (RUNNING)
        OK:      Configuration seems valid
        OK:      Connection to daemon is OK at port 7768
        OK:      Daemon version is: 02.04.01-005_BUILD11.fr
        OK:      Correct connection from arbiter "arbiter-master" ( and no time shift )
        Modules:
            OK:      Name: MongodbRetention        Type: mongodb_retention
```

Checks Shinken

Les checks du pack Shinken permettent de superviser les démons Shinken. Ces checks affichent une pastille SPARE lorsque le démon supervisé est un démon Spare.



Dans le cas d'une défaillance du démon maître, lorsque le Spare prend le relais, une pastille "RUNNING" est rajoutée afin de montrer que le Spare est en cours de fonctionnement :



Conseils d'utilisation

Paramétrage des délais de vérification

Selon votre infrastructure, nous vous conseillons d'ajuster les propriétés **timeout**, **max_check_attempts** et **check_interval**, pour permettre à votre architecture en Haute disponibilité d'être la plus efficace possible. Voir aussi les paramètres de configuration des Démons (*voir la page : Les 7 Démons et 1 script*).

En effet, avec les valeurs par défaut, votre Arbiter vérifie vos différents démons toutes les 60 secondes (***check_interval***). Il considérera le nœud comme "éteint/inaccessible" s'il ne répond pas au bout de 3 secondes (***timeout***) après 3 tentatives de connexion (***max_check_attempts***).

Si vous souhaitez une forte réactivité dans le passage Maître/Spare, vous pourriez par exemple passer votre **check_interval** à 10 secondes et votre **max_check_attempts** à 1. Les délais de bascule seraient alors réduits. Par contre, attention dans un réseau subissant des coupures très régulières ou de fortes latences, car avec ce paramétrage, il se peut que vous basculiez de manière répétée de maître à spare et de spare à maître, et le système de haute disponibilité entraînera de l'instabilité au niveau des communications entre les démons (*multiplication des bascules*). Il en est de même sur de très grosses installations avec des Schedulers et Brokers qui peuvent être surchargés ou subir de fortes latences réseau. Les temps de réponse de disponibilité sont alors plus longs, il faudra adapter vos valeurs.

Il s'agit donc de bien choisir les valeurs selon :

- les latences réseaux entre les démons de votre architecture Shinken,
- votre politique de PRA (*Plan de Reprise d'Activité*).

Pollers spares

> **Remarque sur les Poller spares**
>
> Il est possible comme pour les autres démons d'utiliser des Pollers Spare. Cependant dans le cas du Poller, il serait plus efficace d'utiliser plusieurs Pollers sans utiliser de spare. En effet, les Pollers se répartissent les checks à effectuer de manière transparente. Un Poller Spare inactif peut donc être substitué par un Pollers actif, ce qui est plus intéressant au niveau des performances.