L'Arbiter

Sommaire

Rôle

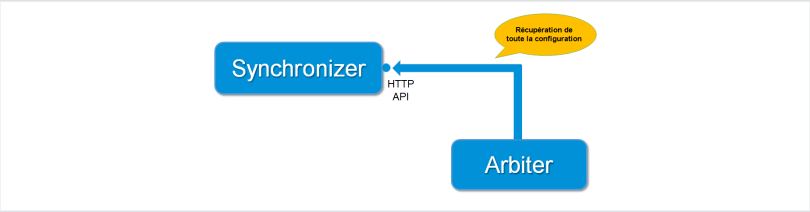
Le démon Arbiter lit la configuration du Synchronizer. Il la divise en N parts (N Schedulers = N parts), et les distribue au démon Shinken Enterprise approprié de manière parallèle. Il gère également la notion de haute disponibilité : si un démon en particulier meurt, il renvoie la configuration gérée par le démon mort à son spare défini qui reçoit les informations des utilisateurs. Il route également les résultats de checks passifs vers le démon approprié. Les résultats de checks passifs sont renvoyés au Scheduler responsable du check.

Important

Il ne peut y avoir qu'un seul Arbiter master et un seul Arbiter spare maximum par infrastructure.

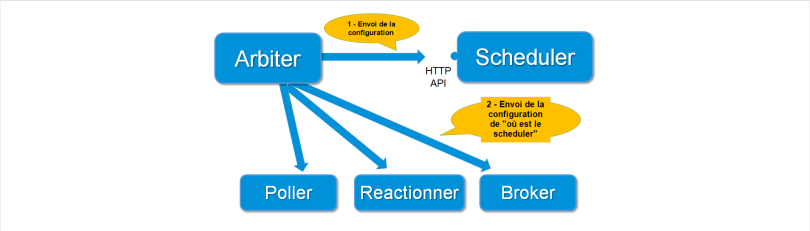
Connexion avec le Synchronizer

La communication entre l'Arbiter et le Synchronizer est faite sur le port standard du Synchronizer (7765).



Connexion avec les autres démons

Ce démon est utilisé pour vérifier et distribuer la configuration aux autres démons, sauf au Synchronizer.
Il se connecte sur le port standard des autres démons, et utilise une connexion HTTP ou HTTPS si ceux-ci l'ont prévue.



Remarque

L'Arbiter est capable de rendre inactifs les démons distants désactivés (lorsque la propriété "enabled" est mise 0 dans leurs fichiers cfg)

Exemple: Depuis le serveur central, je décide de désactiver un Poller distant en passant la propriété "enabled" à 0 dans son fichier de définition. Je redémarre l'Arbiter. Le Poller distant sera affiché comme "désactivé" depuis son shinken-healthcheck local.

Il n'est donc pas nécessaire d'éteindre un démon qui n'est plus utilisé sur une machine distante.

Données

Ce démon stocke la totalité du système de configuration en mémoire. Il a accès à tous les noms de serveurs, adresses, types et commande définis pour les vérifier.

Il stocke également en mémoire les contacts définis, qui doivent recevoir les notifications pour les hôtes et services définis.

Résumé des connexions de l'Arbiter

Démon source	Destination	Port	Protocole	Note
Arbiter	Synchronizer	7765	HTTP/HTTPS	
Arbiter	Scheduler	7768	HTTP/HTTPS	
Arbiter	Poller	7771	HTTP/HTTPS	
Arbiter	Reactionner	7769	HTTP/HTTPS	
Arbiter	Receiver	7773	HTTP/HTTPS	
Arbiter	Arbiter	7770	HTTP/HTTPS	Seulement si il y a un Arbiter esclave, et seulement du maître vers l'esclave.
Arbiter	Broker	7772	HTTP/HTTPS	

Définition du format

Propriété	Défaut	Description
arbiter_name	N/A	Cette variable est utilisée pour identifier le *nom réduit* de l'Arbiter auquel les données sont associées.
host_name	N/A	Cette variable est utilisée par les démons Arbiters pour définir quel objet 'Arbiter' ils sont (de type **maitre** ou **spare**). En effet, tous ces démons sur différents serveurs utilisent la même configuration, donc la seule différence entre eux est le nom du serveur. Cette valeur doit être égale au nom du serveur (comme avec la commande du hostname). Si rien n'est défini pour cette propriété, le démon Arbiter va utiliser le nom du serveur où il est lancé, mais cela ne sera possible qu'avec un seul Arbiter dans l'architecture Shinken.
address	N/A	Cette directive permet de définir l'adresse d'où l'Arbiter principal peut contacter cet Arbiter (qui peut être lui même). Ça peut être un nom DNS ou une adresse IP.
port	7770	Cette directive est utilisée pour définir le port TCP utilisé par ce démon.
use_ssl	0	Cette variable est utilisée pour définir si l'Arbiter doit être contacté en HTTPS (*1*) ou HTTP (*0*). La valeur par défaut est *0* (HTTP).
timeout	3	Définir le temps en secondes que les démons / commande / check de supervision vont attendre que la requête http (PING) pour vérifier que cette Arbiter est vivante aboutisse. Au-delà de ce temps, ce PING sera considéré comme KO. **Elle n'est utilisée que dans les cas suivant**: • Arbiter master qui vérifie que le spare est vivant. • shinken-healthcheck qui récupère les informations de cette Arbiter. • check **Arbiter - KEY - Alive** et **Arbiter - KEY - Performance** qui interroge l'Arbiter. Si ce démon est joignable en HTTPS (use_ssl à 1) avec une latence élevée, nous vous conseillons alors d'augmenter cette valeur de timeout (l'Arbiter aura besoin de plus d'allers/retours pour le contacter).
max_check_attempts	3	**Indique le nombre de tentatives de PING qui doit échouer avec de considérer cette Arbiter comme Mort**. Le paramètre n'est pas utilisé de la même manière si on définit un Arbiter ou un Arbiter spare: • Si vous êtes dans un Arbiter, cette valeur sera utilisée seulement par l'Arbiter spare pour calculer le temps d'attente avant de prendre la main sur l'Arbiter s'il n'a pas eu de connexion de ce dernier (*max_check_attempts* * *check_interval* seconds), car l'Arbiter spare n'essaye jamais de contacter l'Arbiter (pour des raisons de sécurité) • Si vous êtes dans un Arbiter spare, cette valeur sera utilisée par l'Arbiter comme le nombre de tentatives de PING à faire (qui auraient donc échoués) avant de considérer l'Arbiter spare comme mort.
check_interval	60	**Indique le temps d'attente entre les tentatives de PING (Exemple: si la première tentative échoue, la suivante aura lieu après *checks_interval* secondes)** • Si vous êtes dans un Arbiter, cette valeur sera utilisée seulement par l'Arbiter spare pour calculer le temps d'attente avant de prendre la main sur l'Arbiter s'il n'a pas eu de connexion de ce dernier (*max_check_attempts* * *check_interval* seconds), car l'Arbiter spare n'essaye jamais de contacter l'Arbiter (pour des raisons de sécurité) • Si vous êtes dans un Arbiter spare, cette valeur sera utilisée par l'Arbiter comme délai entre le nombre de tentatives de ping à faire (qui auraient donc échoués) avant de considérer l'Arbiter spare comme mort.
spare	0	Cette variable permet de savoir si le démon correspondant à la définition de l'Arbiter est un spare ou pas. La valeur par défaut est *0* (maître/non-spare).
modules	N/A	Cette variable définit tous les modules chargés par l'Arbiter qui correspond à cette définition.

data_timeout	120	Cette variable est utilisée pour définir le temps en secondes avant de considérer un transfert de configuration vers l'Arbiter spare comme échoué.
enabled	N/A	Cette variable est utilisée pour définir si l'Arbiter est activé ou non.
mismatch_version_error	1	Cette variable est utilisée pour définir le comportement en cas de version différentes entre l'arbiter et les autres daemons, Si l'option est activée, les daemons refusent la configuration de l'arbiter et une erreur est remontée, Si l'option est désactivée, les daemons acceptent la configuration de l'arbiter et un warning est remonté.

Exemple de définition

Dans le répertoire **/etc/shinken/arbiters/**, voici un exemple de définition qui permet la définition de l'Arbiter (à placer dans un fichier .cfg) :

⚠️ Il est conseillé d'éditer les fichiers .cfg avec l'encodage utf-8

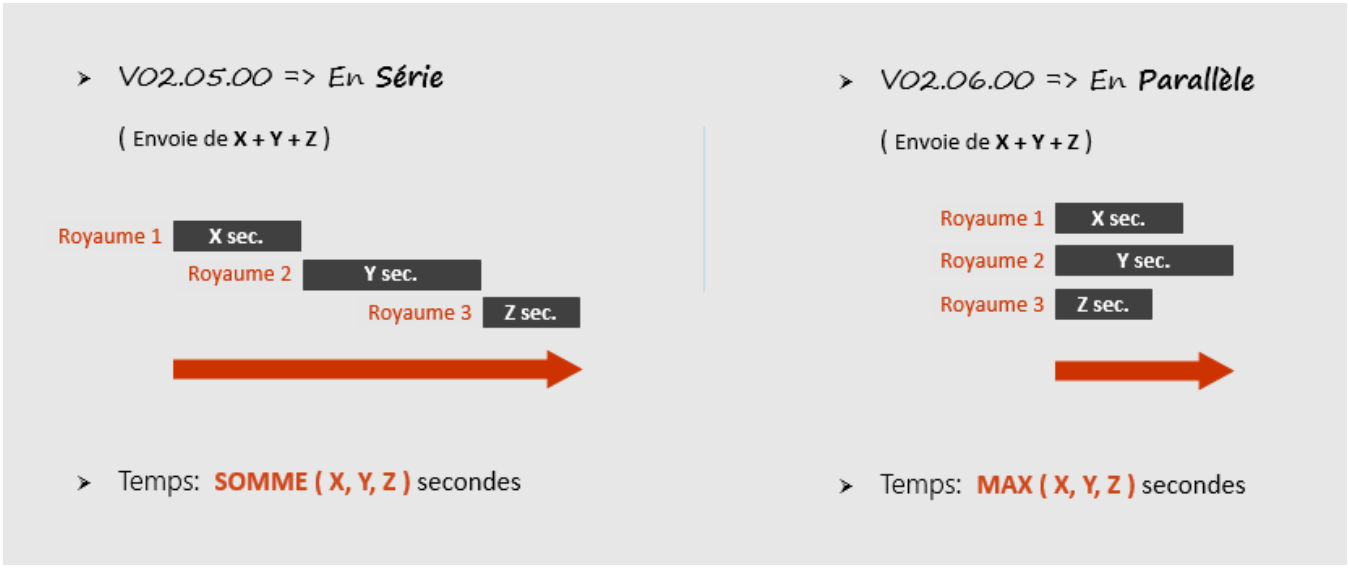
```
#================================================================================
# ARBITER
#================================================================================
# Description: The Arbiter is responsible for:
# - Loading, manipulating and dispatching the configuration
# - Validating the health of all other Shinken daemons

#================================================================================
# IMPORTANT: If you use a spare arbiter you MUST set the host_name on each
# servers to its real DNS name ('hostname' command).
#================================================================================

define arbiter {

    #======== Daemon name and address =========
    # Daemon name. Must be unique
    arbiter_name      arbiter-master

    # Hostname used by the Arbiter daemon to distinguish master and slave Arbiters (when having multiple
Arbiters).
    # If not defined, the Arbiter daemon will use its server name, but it can be used with only one Arbiter
in the
    # Shinken architecture.
    #host_name        HOSTNAME

    # IP/fqdn of this daemon
    address           localhost   ; DNS name or IP

    # Port (HTTP/HTTPS) exposed by this daemon
    port              7770

    # 0 = use HTTP, 1 = use HTTPS
    use_ssl                 0


    #======== Daemon connection timeout and down state limit =========
    # Parameters used by Arbiter-master and Arbiter-spare to put respectively the other demon as not
available.

    #---------------
    # timeout: value specific to each arbiter. When trying to PING this Arbiter, any daemon / healtcheck /
shinken checks will consider it as unreachable after this timeout
    # All cases:
    #   - Arbiter master wait the arbiter spare for this timeout duration
    #   - shinken-healtcheck/shinken check will use it to wait for the arbiter master or spare
    #   ( default 3 )
    # NOTE: this parameter is NOT used by the arbiter spare to wait for master's death before
    #       take over the configuration. For this, look at max_check_attempts & check_interval
    timeout             3

    #---------------
    # data_timeout: how many second to consider a configuration transfert to be failed to the arbiter spare
    #   - to be defined on the Arbiter-spare
    #   - Used by the Arbiter-master
    #   ( default 120 )
    data_timeout        120
```

```
    #---------------
    # max_check_attempts: how many fail check to consider this daemon as DEAD
    #   Defined on the Arbiter-spare
        #     -> Used by the Arbiter-master to declare the spare as DEAD ( so after a wait of X
max_check_attempts * check_interval seconds )
    #
    #   or
    #
    #   Defined on the Arbiter-master
        #     -> Used by the Arbiter-spare to take over the role of arbiter-master ( so after a wait of X
max_check_attempts * check_interval seconds )
    #
    max_check_attempts   3

    #---------------
    # check_interval: check every X seconds or use to know how many seconds to wait for the arbiter master
    #   Defined on the Arbiter-spare
        #     -> Used by the Arbiter-master to declare the spare as DEAD ( so after a wait of X
max_check_attempts * check_interval seconds )
    #
    #   or
    #
    #   Defined on the Arbiter-master
        #     -> Used by the Arbiter-spare to take over the role of arbiter-master ( so after a wait of X
max_check_attempts * check_interval seconds )
    #
    check_interval       60


    #======== Master or spare selection =========
    # 1 = is a spare, 0 = is not a spare
    spare             0


    #======== Modules to enable for this daemon =========
    # Available:
    # - synchronizer-import : [mandatory] will allow to get configuration from the synchronizer
    # - architecture-export : will open a socket that listens for Shinken architecture descriptions to
generate corresponding hosts and NagVis maps
    modules                 synchronizer-import


    #======== Enable or not this daemon =========
    # 1 = is enabled, 0 = is disabled
    enabled       1


    #======== Enable or not this daemon =========
    # 1 = is enabled, 0 = is disabled
    # enabled:  if the version between the arbiter and daemons mismatch, daemons refuse the arbiter
configuration and an error is reported
    # disabled: if the version between the arbiter and daemons mismatch, daemons accept the arbiter
configuration and a warning is reported.
    mismatch_version_error 1
}
```

Remarques sur la compilation et l'envoi de la configuration

Un des rôles de l'Arbiter est de récupérer la configuration des démons, hôtes et checks et de la transmettre aux démons appropriés.

Lorsque la configuration comporte un nombre important d'hôtes et de checks, la compilation de la configuration peut être longue. De même, si on dispose d'une architecture comportant plusieurs royaumes, il est possible que le démon de certains royaumes soit situé dans des zones géographiques ou configurations réseau ou les débits et latences de connexion sont élevés. Ces connexions réseau moins optimisées peuvent occasionner des ralentissements sur l'ensemble des royaumes lors de l'envoi de la configuration des hôtes/checks vers les démons.



Pour limiter l'impact de ces problèmes, depuis la V02.06.00, l'Arbiter effectue la compilation et l'envoi de la configuration de manière parallèle pour chaque royaume. Un ralentissement occasionné par une connexion ralentie vers un royaume n'aura alors pas d'effet sur l'envoi de la configuration aux autres royaumes de l'architecture Shinken.