

Le Broker

Rôle

Le démon broker exporte et gère les données du Scheduler. Sa gestion ne peut se faire qu'à travers des modules. Plusieurs modules de gestion peuvent être activés en même temps.

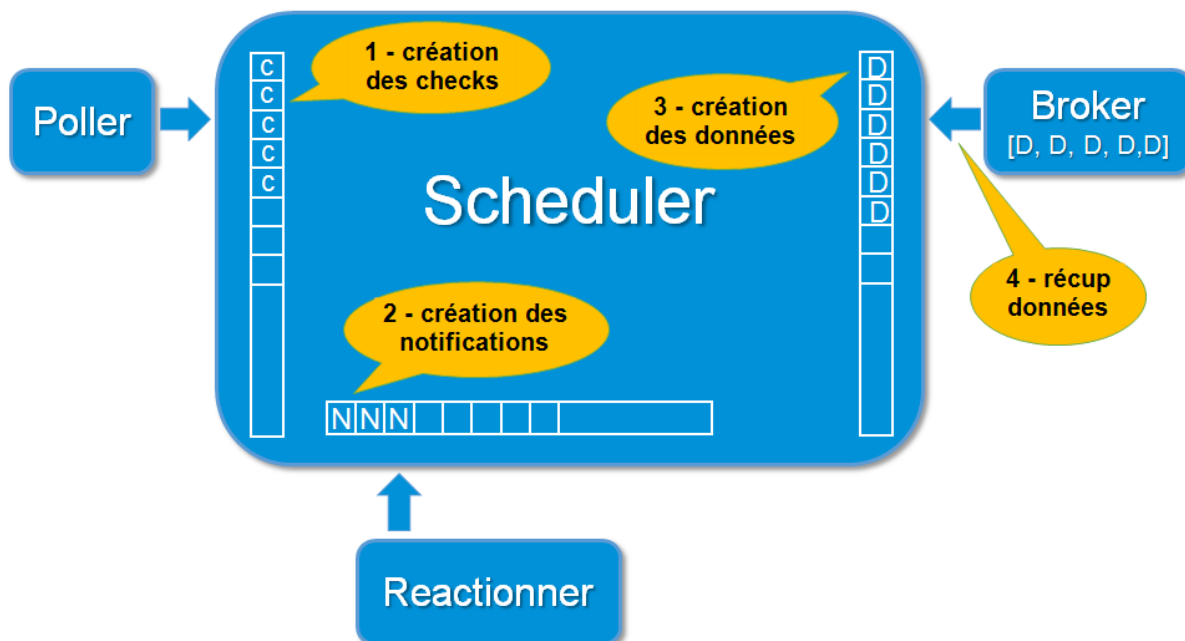
Exemples de modules du Broker :

- Module pour centraliser les logs Shinken : Simple-log (flat file)
- Module pour exporter les données: Graphite-Perfdata
- Module pour l'API Livestatus
- Module pour l'affichage de l'interface de visualisation : WebUI

Données

Le Broker reçoit toutes les données des Schedulers. Il garde également en mémoire les données des hôtes et des checks.

Enfin, il sauvegarde les résultats des checks dans une base mongodb (si possible, cette base doit être installée sur le même serveur que le broker).



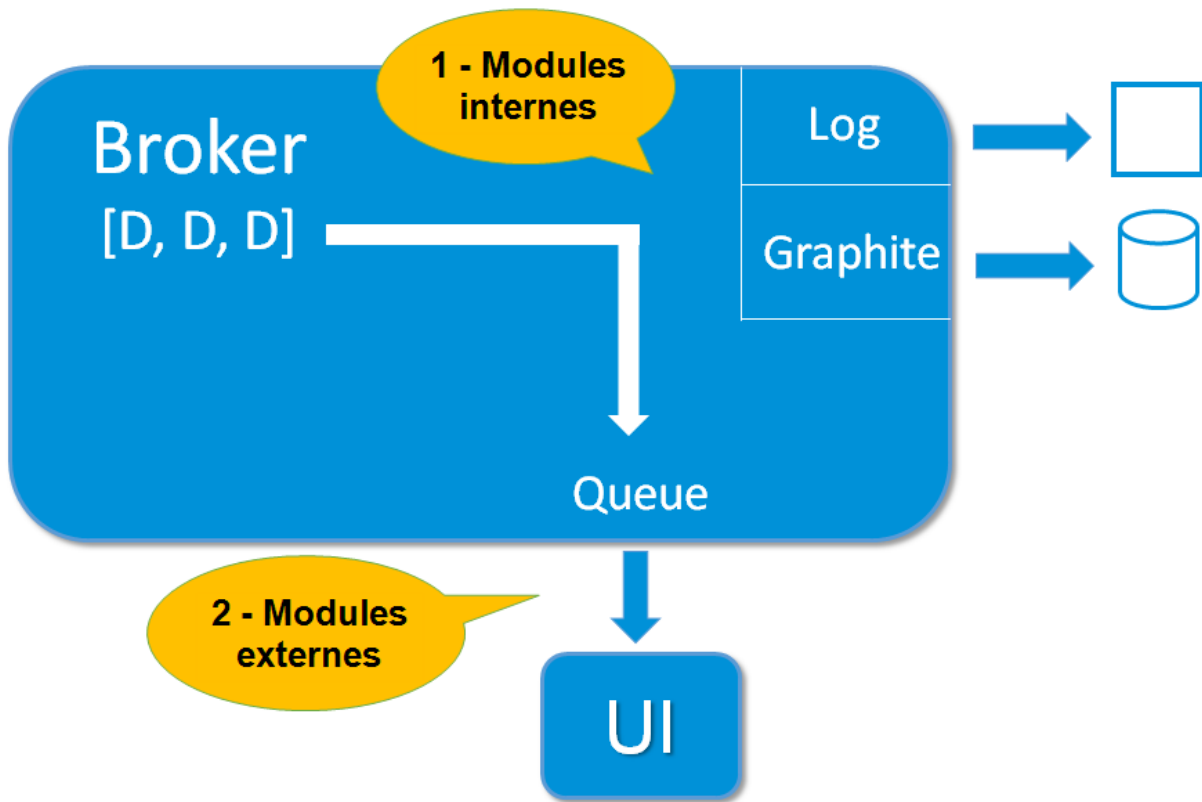
Données de métrologie

Les données de métrologie sont sauvegardées sur le serveur du broker dans l'application graphite.

Cette application écoute le port 2003, et cette connexion se fait sans authentification.

Cette application doit donc écouter exclusivement sur l'interface réseau locale (loopback) du serveur du broker.

Logique interne du Broker



Résumé des connexions du Broker

Source	Destination	Port	Protocole	Note
Broker	Scheduler	7768	HTTP/HTTPS	
Broker	Poller	7771	HTTP/HTTPS	
Broker	Reactionner	7769	HTTP/HTTPS	
Broker	Receiver	7773	HTTP/HTTPS	
Broker	Broker (local)	2003	TCP	interface localhost uniquement

Description des variables

Propriété	Défaut	Description
broker_name	N/A	Cette variable est utilisée pour identifier le *nom réduit* du Broker auquel les données sont associées.
address	N/A	Cette directive est utilisée pour définir l'adresse permettant de joindre ce Broker. Par défaut "localhost", changez le par un nom DNS ou une adresse IP.
port	7772	Cette directive est utilisée pour définir le port TCP utilisé par le démon.
use_ssl	0	Cette variable est utilisée pour définir si le Broker doit être contacté en HTTPS (*1*) ou HTTP (*0*). La valeur par défaut est *0* (HTTP).
spare	0	Cette variable est utilisée pour définir si le broker peut être géré comme un spare (prendra uniquement la configuration si le maître échoue). La valeur par défaut est *0* (maître).

timeout	3	Cette variable est utilisée pour définir le temps en secondes avant que l'Arbiter ne considère ce démon comme à l'arrêt. 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).
data_timeout	120	Cette variable est utilisée pour définir le temps en secondes avant de considérer un transfert de configuration ou de données comme échoué.
max_check_attempts	3	Si le ping permettant de détecter la disponibilité réseau du nœud est en échec N fois ou plus, alors le nœud est considéré comme mort. (par défaut, 3 tentatives)
check_interval	60	Intervalle de Ping toutes les N secondes.
modules	N/A	Cette variable est utilisée pour définir les modules chargés par le broker.
realm	N/A	Cette variable est utilisée pour définir le royaume où le broker doit être. Si aucun n'est sélectionné, celui par défaut lui sera assigné.
manage_sub_realms	1	Cette variable est utilisée pour définir si le broker prendra des tâches des Schedulers des sous-royaumes .
manage_arbiters	1	Prend les données de l'arbiter. Il ne devrait y avoir qu'un seul broker pour l'arbiter.
satellitemap	N/A	Cette variable est utilisée pour définir, pour des environnements NATés, les différents satellites comme vus depuis ce broker.
enabled	N/A	Cette variable est utilisée pour définir si le broker est activé ou non.

Définition - exemple

Dans le répertoire `/etc/shinken/brokers/`, voici un exemple de définition qui permet la définition du Broker (à placer dans un fichier CFG) :

```

=====
# BROKER
=====
# Description: The broker is responsible for:
# - Exporting centralized logs of all Shinken daemon processes
# - Exporting status data
# - Exporting performance data
# - Exposing Shinken APIs:
#   - Status data
#   - Performance data
#   - Command interface
=====

define broker {

    #===== Daemon name and address =====
    # Daemon name. Must be unique
    broker_name          broker-1

    # IP/fqdn of this daemon (note: you MUST change it by the real ip/fqdn of this server)
    address              nodel.mydomain

    # Port (HTTP/HTTPS) exposed by this daemon
    port                 7772

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

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

    #===== Daemon connection timeout and down state limit =====
    # timeout: how many seconds to consider a node don't answer
    timeout              3

```

```
# data_timeout: how many second to consider a configuration transfert to be failed
# because the network bandwidth is too small.
data_timeout          120

# max_check_attempts: how many fail check to consider this daemon as DEAD
max_check_attempts    3

# Check this daemon every X seconds
check_interval        60

#=====  
# Available:  
# - Simple-log          : save all logs into a common file  
# - WebUI              : visualisation interface  
# - Graphite-Perfdata   : save all metrics into a graphite database  
# - sla                : save sla into a database  
# - Livestatus         : TCP API to query element state, used by nagios external tools like NagVis  
or Thruk  
modules              Simple-log, WebUI, Graphite-Perfdata, sla

#=====  
# Realm and architecture settings =====  
# Realm to set this daemon into  
realm                All

# 1 = take data from the daemon realm and its sub realms  
# 0 = take data only from the daemon realm  
manage_sub_realms    1

# Is enabled, then this broker will receive data (logs and) from the arbiter  
manage_arbiters      1

# In NATted environments, you declare each satellite ip[:port] as seen by  
# *this* broker (if port not set, the port declared by satellite itself  
# is used)  
#satellitemap        scheduler-1=1.2.3.4:7768, poller-1=1.2.3.5:7771

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

}
```