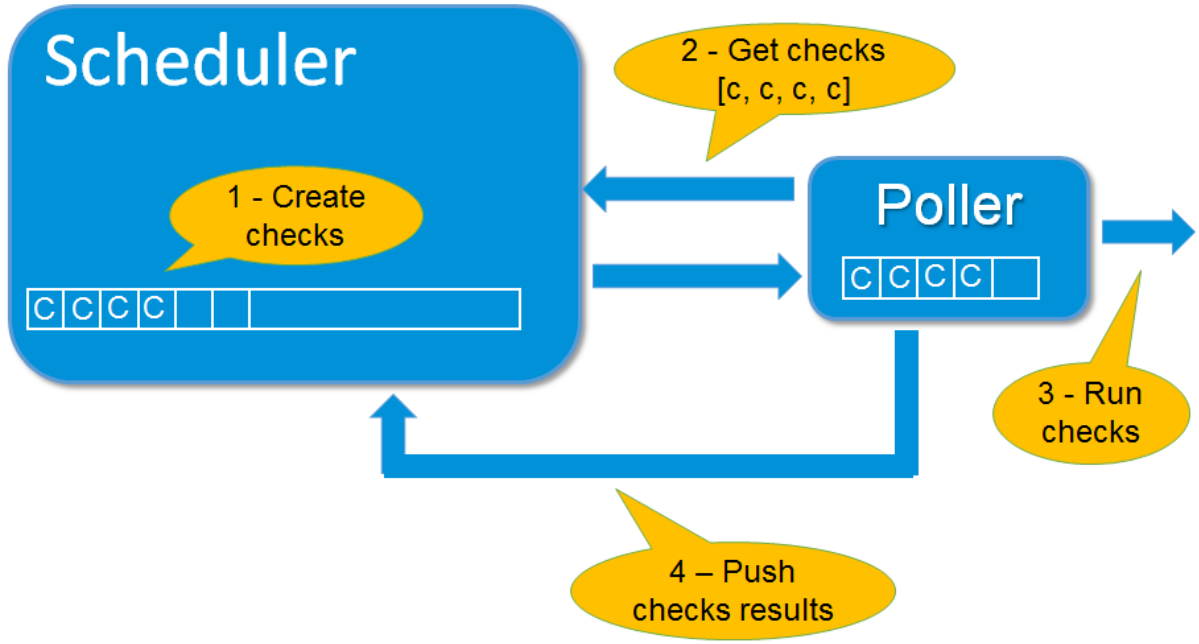
Rôle

Le démon poller lance les sondes tel que demandé par les schedulers. Quand la vérification est terminée, il renvoie le résultat aux schedulers. Les pollers peuvent être définis pour des vérifications spécifiques (ex. Windows versus Unix, clientr A versus client B, DMZ). Il peut y avoir plusieurs pollers pour des questions de load-balancing .

Connexions avec les autres démons

Le poller récupère sa configuration depuis l'arbiter , par défaut sur son port 7771 .

Sa configuration correspond à la liste de royaumes du scheduler auquel il devra se connecter .

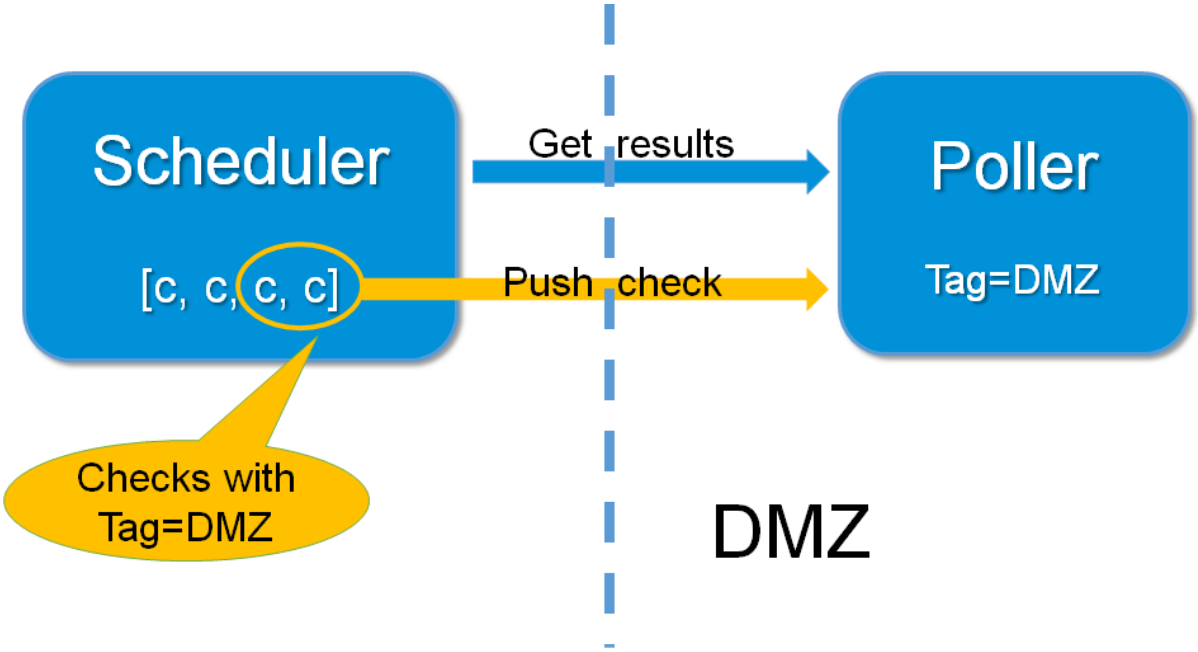


DMZ Poller

Il est possible de définir un poller en DMZ , et les schedulers dans le LAN. Dans ce cas, l'objectif est d'éviter les connections de DMZ (poller) vers LAN (schedulers) .Il est alors possible de définir le poller comme passif. Le scheduler va alors ouvrir une connexion vers le poller.

C'est le réglage recommandé pour une DMZ.

Seuls les checks pour les hôtes dans la DMZ seront lancés par ce poller.

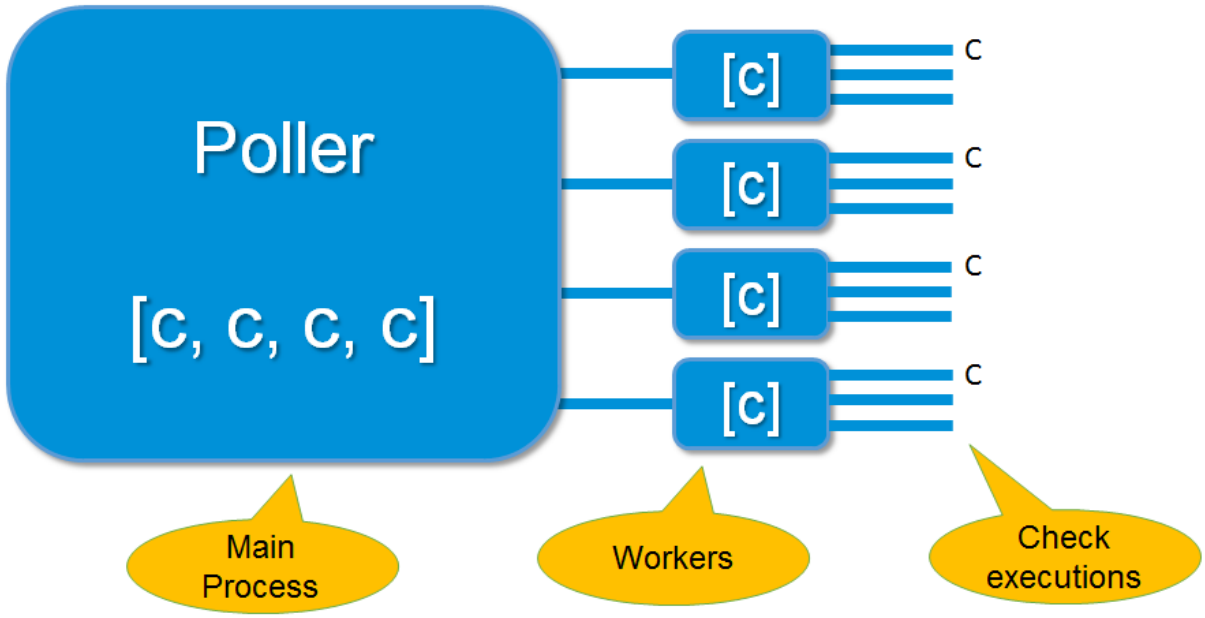


Données

Le poller reçoit la commande des schedulers.

Il est important de noter que les sondes lancés par le poller auront un acès direct aux hôtes et devront récupérer des données venant de ceux-ci.

Poller Internes



Résumé des connexions du poller

source	Destination	Port	Protocole	Note
Poller	Scheduler	7768	HTTPS	

Différents types de Pollers: poller_tag

L'architecture actuelle de Shinken Enterprise architecture est très utile pour une organisation qui utilise le même type de poller pour tous les checks. Mais il peut être nécessaire d'avoir différents types de poller.

Cela peut être utile quand l'utilisateur a besoin d'avoir ses hôtes dans le même scheduler (comme avec les dépendances) mais a besoin d'avoir des hôtes ou des checks vérifiés par un poller spécifique.

Ces vérifications peuvent être qualifiés à 3 niveaux :

- Hôte
- Check
- Commande

Le paramètre pour qualifier une commande, un hôte ou un check est le "poller_tag".

L'existance du paramètre est détermité dans l'ordre de cascade suivant:

- On regarde d'abord sur la commande
- puis le check
- puis sur l'hote.

Les pollers peuvent être identifié par plusieurs poller_tags. Si ils ont des tags, ils ne prendront que les checks qui correspondent à ce tag.

C'est principalement utilisé dans un réseau en DMZ, vous avez besoin d'un poller dédié dans cette DMZ qui retourne les résultats au scheduler dans le LAN. Dans ce cas, vous pouvez toujours avoir des dépendances entre les hôtes dans la DMZ et le LAN, et être certain que les checks ne sont effectués que par le poller dans la DMZ.

Descriptions des variables

Propriété	Défaut	Description
poller_name	N/A	cette variable est utilisée pour définir le nom raccourci du poller auquel sont rattachées les données .
address	N/A	Cette directive est utilisée pur définir l'adresse d'où l'arbiter principal peut joindre le poller. ça peut être un nom DNS ou une adresse IP .
port	7771	Cette directive est utilisée pour définir le port TCP utilisée par ce démon .
spare	0	Cette variable est utilisée pour définir si le poller peut être géré comme spare (ne chargera la configuration que si le maître tombe)
realm	N/A	Cette variable est utilisée pour définir le royaume ou sera mis le poller . SI aucun n'es spécifié, il sera rattaché au royaume par défaut.
manage_sub_realms	0	Cette variable est utilisée pour définir si le poller acceptera les tâches du scheduler provenant des sous-royaumes de son royaume.
poller_tags	None	Cette variable est utilisée pour définir les checks que peut lancer le poller. Si il n'y a aucun poller_tags spécifié, il prendra tous les checks non qualifiés. Par défaut, il n'y a aucun poller_tag, donc il prend tous les checks non qualifiés. .
modules	N/A	Cette variable est utilisée pour définir tous les modules que le scheduler va charger. .

Exemple de définition

```
define poller{
   poller_name           Europe-poller
   address               node1.mydomain
   port                  7771
   spare                 0
   manage_sub_realms     0
   poller_tags           DMZ, Another-DMZ
   modules               module1,module2
   realm                 Europe
   min_workers           0 ; Starts with N processes (0 = 1 per CPU)
   processes_by_worker   256 ; Each worker manages N checks
   polling_interval      1 ; Get jobs from schedulers each N seconds
}
```