Supervision et métrologie

Philippe GRÉGOIRE

Supervision : définition

- Supervision:
  - Surveillance du bon fonctionnement d'un système

Métrologie : définition

- Métrologie:
  - Du grec metron – la mesure
  - Ensemble d'outils permettant de mesurer,
    - associer des valeurs à une observation.

Supervision et métrologie: pourquoi

- Être alerté en temps réel
  - Complexité matérielles des clusters
    - Switchs, réseaux, serveurs, cpu, mémoire
  - Complexité logicielles des clusters
    - Imbrication de nombreux services internes et externes
  - Nécessité d'une architecture robuste
    - Éviter les SPOFs
    - Architecture redondante sur les points critiques
  - Certitude d'une panne
    - Matérielle (MTBF)
    - Logicielle : impact des mises à jour

Supervision et métrologie: pourquoi

- Retrouver l'origine du problème
  - Effet domino:
    - Inter-dépendances des couches logicielles et matérielles.
    - Un dysfonctionnement d'un composant engendre rapidement l'apparition d’autres problèmes
    - Exemples:
      - Remplissage du disque système sur un serveur
      - Surcharge du serveur d'identification
  - Nécessité de conserver l'historique des alertes
    - Permet d'avoir une vue d'ensemble
    - Analyse à posteriori
    - Corrélation entre des événements

Supervision et métrologie: pourquoi

- Surveiller en l'absence de personnel
  - Centre de calcul 24/24 , 7 jours sur 7
  - Pas d'utilisateurs pour reporter des dysfonctionnements
  - Pas d'administrateurs pour surveiller
  - Système de surveillance couplé à un système d'alertes (personnel sous astreinte)

Supervision et métrologie: pourquoi

- Anticiper les problèmes
  - Agir sur avertissements
    - Ne pas attendre les périodes d'absence,
    - Ne pas attendre les situations bloquantes,
    - Ne pas attendre les situations désespérées.

Supervision et métrologie: pourquoi

- Respect des engagements contractuels
  - Engagements du constructeur
  - Engagements vis à vis de ses propres clients
- Être capable de communiquer/expliquer
  - Communiquer lors des pannes, juste après.
  - Rendre compte.

Supervision et métrologie: pourquoi

- Assurer le bon fonctionnement des services
  - Mesurer la disponibilité du service
  - Mesurer la performance des services
    - Disposer de valeurs objectives de fonctionnement
    - Ressenti des utilisateurs
- Améliorer le fonctionnement des services
  - Identification des goulots d'étranglement
  - Identification des maillons faibles
- Assurer le bon dimensionnement (métrologie)
  - Optimisation des ressources et des coûts.

Supervision et métrologie: quoi

- Périmètre de la supervision
  - Le cluster (et quels composants?)
  - Les serveurs du centre de calcul
  - La salle machine ? Les infrastructures ?
    - Sas d'accès aux salles
    - Groupes électriques
    - Systèmes de climatisation
- Périmètre déterminé par :
  - les contraintes de sécurité (réseau spécifique pour infra)
  - Les contraintes d'organisation : intervenants, locaux, accès.

Supervision et métrologie: comment

- Être alerté comment ?
  - Les utilisateurs ?
    - Pas en temps réel (quoi que), pas toujours là (centre de calcul)
    - Informations non pertinentes (ça ne marche plus !)
    - Perte de temps (user & admin)
    - Dégradation du service rendu
    - Quid des ressources utilisées occasionnellement ?
      - Éviter les situations d'urgence
- Nécessité d'un système autonome de surveillance
  - Automatique
  - Continu
  - Couplé à un système d'alerte

Supervision : méthodologie

- Définir le périmètre:
  - Cluster : nœuds et services
  - Serveurs et services
  - Criticité : services sous astreinte
  - Périmètre supervision # périmètre métrologie
- Avoir une vue d'ensemble de l'architecture
  - Du cluster
  - Du centre de calcul
  - Interdépendances services / services , services / matériels
  - Hiérarchie de surveillance

Supervision : méthodologie

- Mise en place progressive:
  - Commencer par un cercle restreint
  - Tester
  - Mesurer l'impact
- Continue et évolutive
  - Agrandir le périmètre,
  - Affiner en fonction des incidents
- Observer régulièrement le système en production
  - Profile temporel d'utilisation
  - Charge normale des services

Supervision : méthodologie

- Capitaliser l'expérience
  - Utiliser un outil de suivi d'incidents (Mantis)
  - Utiliser une base de connaissance (Wiki)
    - Prise d'informations
    - Résolution rapide des incidents
  - Mettre en place des procédures
    - Retour rapide en condition normale
- Partage de connaissances
  - Amélioration des temps de résolution.
  - Gain de temps pour les administrateurs

Supervision : alertes

- Donner les bonnes informations :
  - Être synthétique
    - Objet de l'alerte
    - Criticité
  - Être clair
    - Utiliser le bon vocabulaire

Supervision : alertes

- Aux bonnes personnes :
  - Trop d'informations tue l'information
  - Pas d'affolement
  - Envoyer uniquement aux personnes impactées

Supervision : alertes

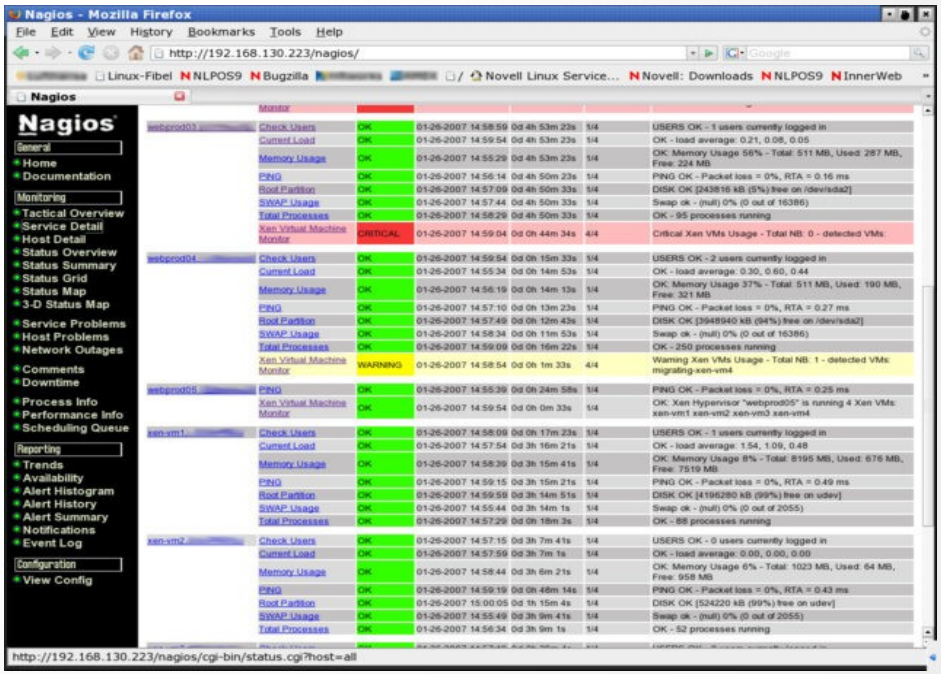
- Sans les noyer :
  - Envoyer un seul message sur le problème à la source
  - Filtrer les alertes sur les problèmes qui en découlent
  - Éviter les effets domino/cascade

Métrologie : indicateurs

- Choisir les bons indicateurs
  - Les indicateurs représentatifs
  - Le bon nombre
- Évaluer le volume journalier, mensuel
  - Cluster de 4000 nœuds, 100 mesures /min 1T/mois
  - Durée et politique de conservation des données
- Évaluer l'impact de la collecte
  - Les nœuds
  - Le réseau
  - Les serveurs de stockage.

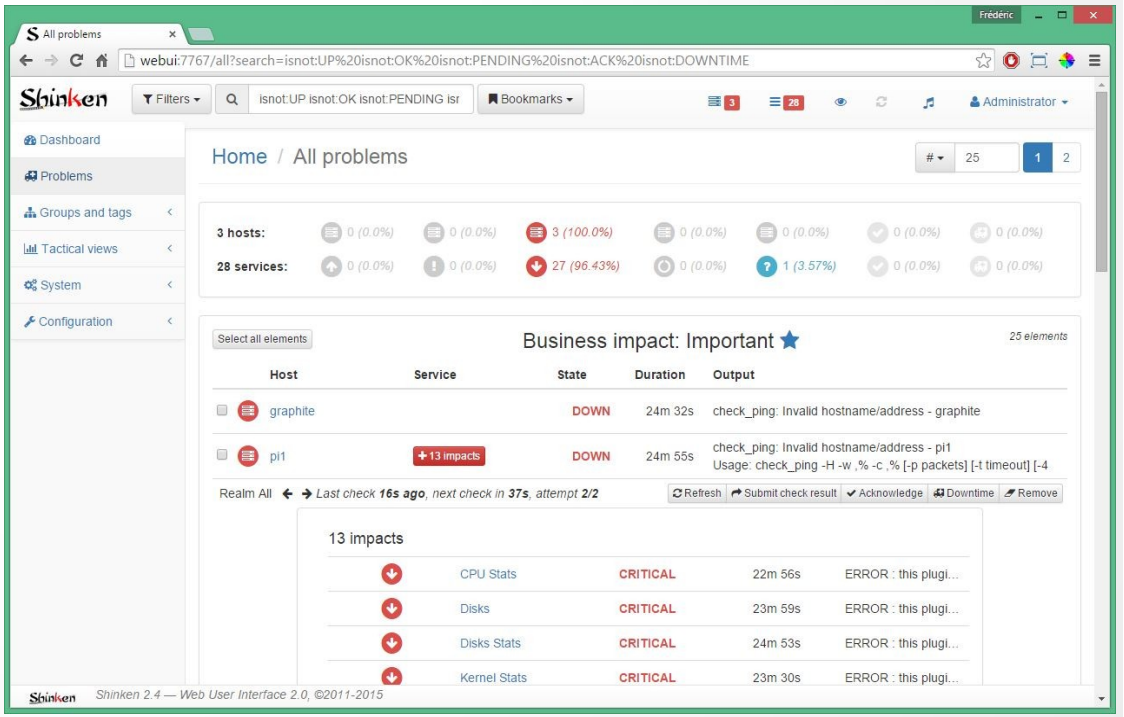
Supervision : Nagios

- Leader historique des solutions OpenSource
- Large communauté
- Écosystème de plugins de vérification
- Possibilité de modules d'extension
- Bonne documentation
- Performances
- GUI un peu vieille
- Développement



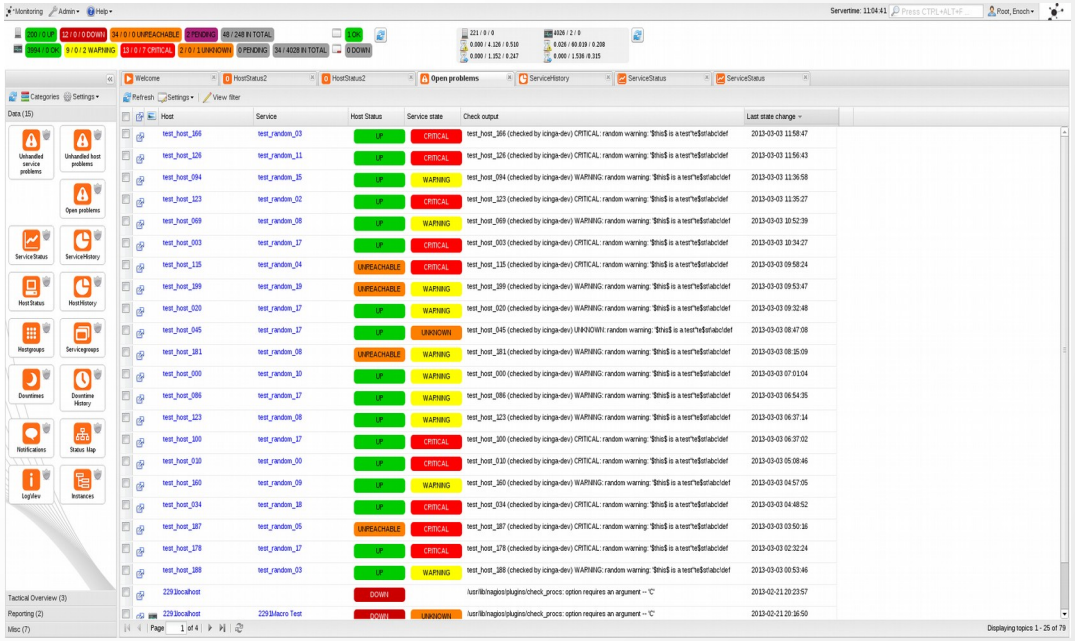
Supervision : Shinken

- Ré-implémentation OpenSource de Nagios en python
- Architecture distribuée et extensible
- Compatibilité avec Nagios (config et plugins)
- Développement actif
- Bonne documentation
- Performances
- GUI



Supervision : Icinga

- Ré-implémentation OpenSource de Nagios en C++
- Connectivité avec MySQL, Oracle, PostGres
- Rest API pour intégration facile
- Compatibilité avec Nagios (config et plugins)
- Bonne documentation
- Performances
- GUI WEB2
- Développement actif

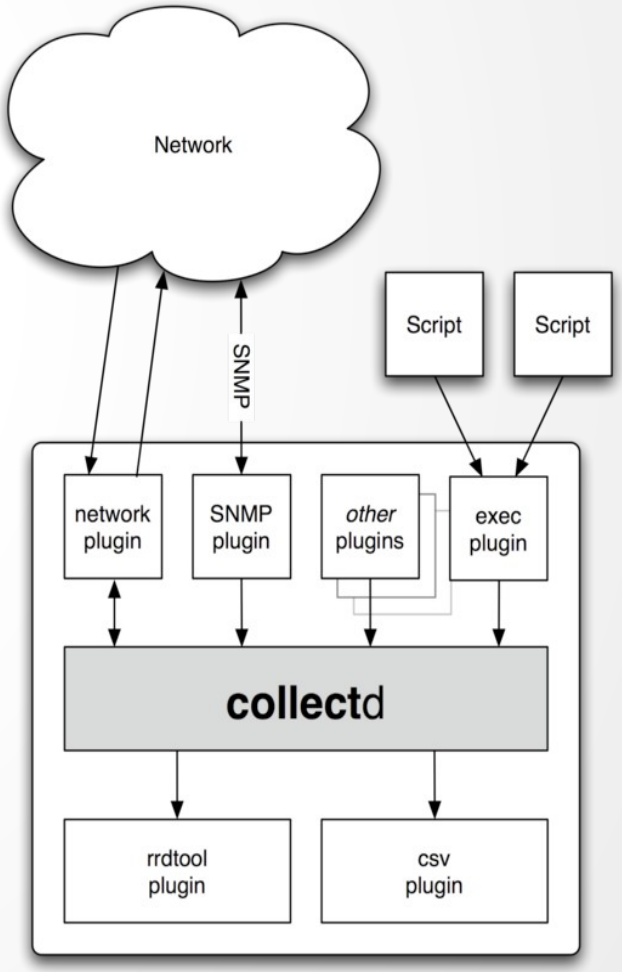


Métrologie : Collectl

- Outil de collecte de données
- Prise de temps calée sur les secondes et sans dérive
  - Analyse sur un cluster
- Surcharge très faible
- Agrégation de données: ex somme charge cpu.
- Affichage des données synthétique ou détaillé
- Sortie en mode plot (CSV)
- Écriture des données dans un fichier ou une socket
- Interface avec Ganglia ou Graphite
- Beaucoup de métriques : cpu, disk , mémoire, infiniband, lustre
- 6000 lignes de code Perl, facile à modifier, pas modulaire

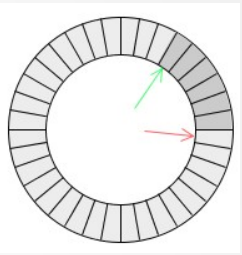
Métrologie : Collectd

- Outil de collecte de données écrit en C
- Modulaire
- Surcharge très faible
- Sortie en mode plot (CSV) ou RRD
- Implémentation réseau riche :
  - unicast, multicast, ipv6, proxy
- Beaucoup de plugins :
  - RRD, Graphite,
- Seuils et notifications

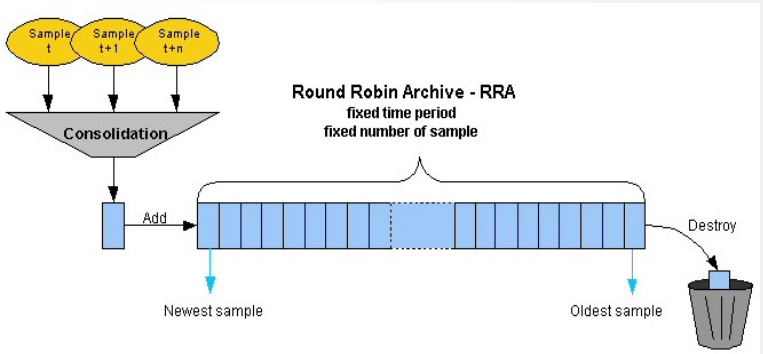


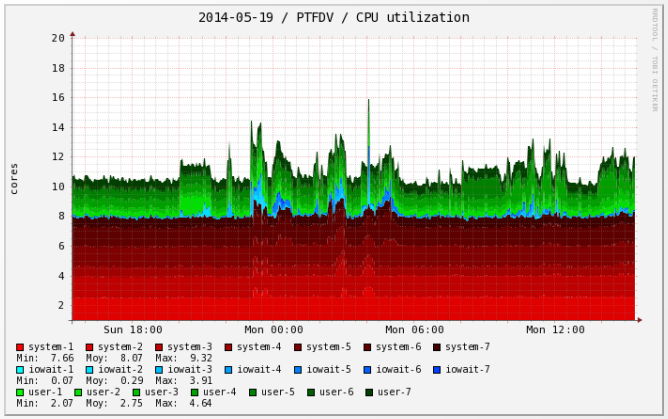
Métrologie : RoundRobinDatabase

RRDtool
logging & graphing



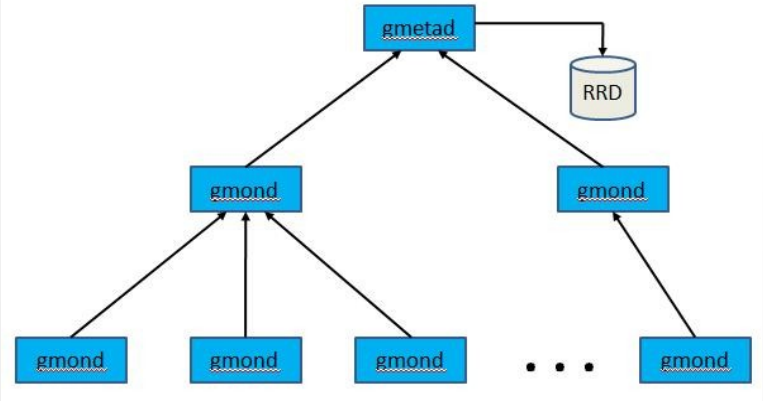
- Outil de gestion de données chronologiques
- Bases de **taille fixe**, la sauvegarde est cyclique.
- Outils de création de base
- Consolidation temporelle des données collectées
  - Permet de changer d'échelle de temps
    - Ex : de 1 valeur /min à 1 valeur / 5min.
  - Une fonction : max, min, somme, moyenne,
  - Un intervalle de consolidation
  - Un nombre maximum de valeurs
- Multi sources
- Tendances
- Outils de tracé de graphiques
- Outils d'export en XML, texte
- Charge IO
  - rrdcached
- Perte de granularité :
  - Impossible de retrouvrer des anciennes valeurs après consolidation





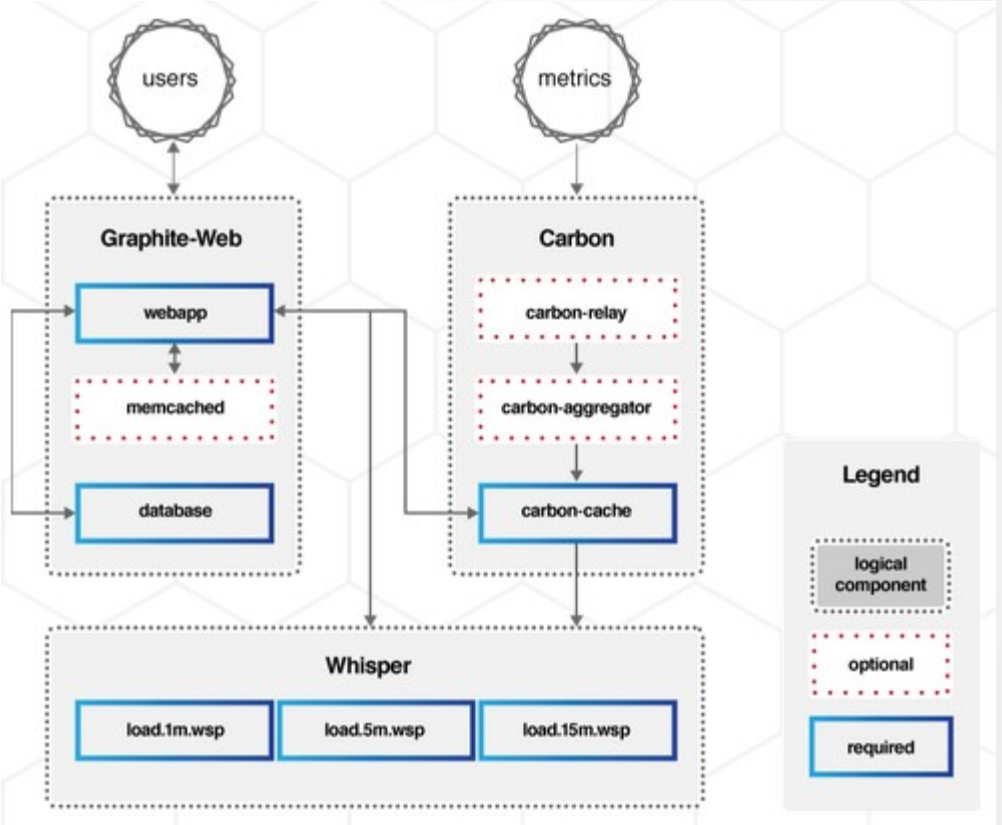
Métrologie : Ganglia

- gmond : service sur chaque nœud surveillé collectant les différentes métriques.
- gmetad : service consolidant les données des services gmond et les stockant dans une base RRD.
- Extensible : 2000 nœuds
- Interface Web



Métrologie : Graphite

- Outil d'enregistrement de données et de graphes.
- carbon – service de réception de données chronologiques.
  - Gestion de cache
  - Écriture asynchrone dans une base de données
- whisper – base de données ~ RRD
- graphite-web – GUI & API pour créer des graphes et tableaux de bord.
  - Fonctions mathématiques
  - Métacaractères :
    - cluster.nodes.*.loadavg



Métrologie : Graphana

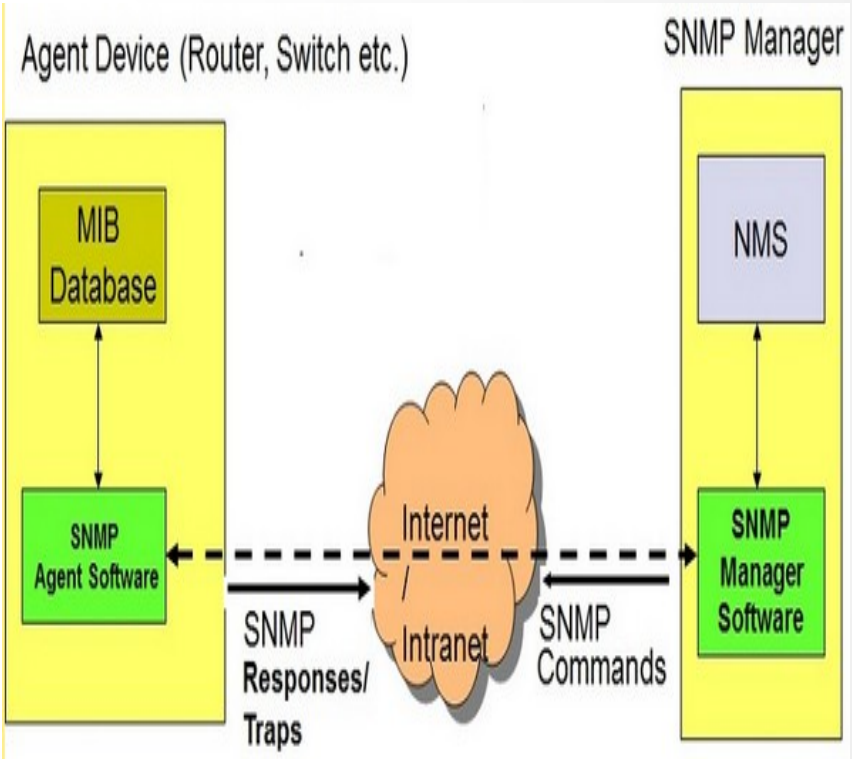
- Outil d'analyse et de visualisation de données
- Ne stocke pas les données, accepte plusieurs sources :
  - Graphite, MySQL, Promotheus,
- Accès aux données via des Organisations/Users
- Tableaux bâtis à partir de panels : Graph, Singlestat, Dashlist, Table,and Text
- Requêtes dynamiques

SNMP

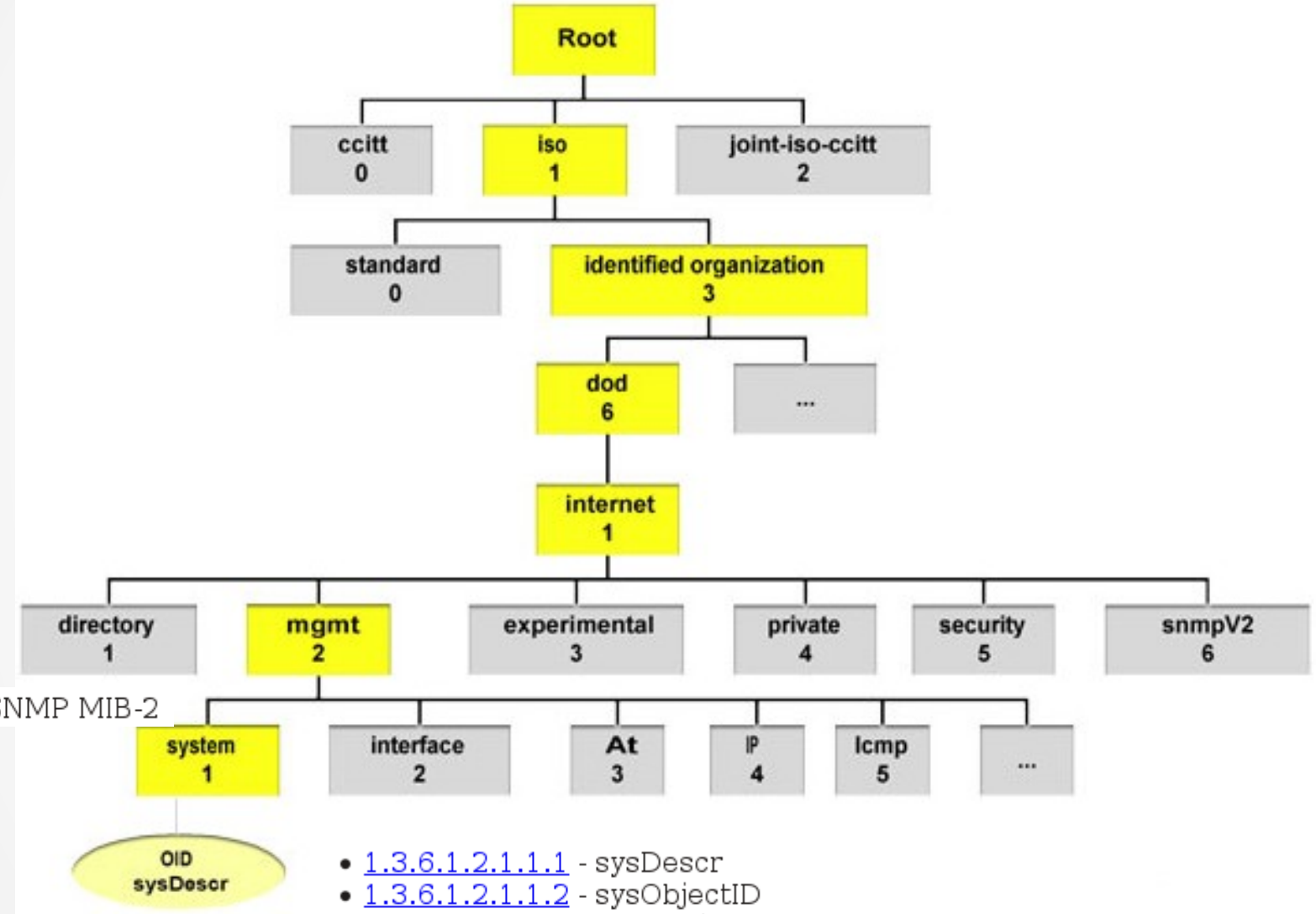
- Simple Network Management Protocol
- Protocole de communication
- Gestion et supervision d'équipements réseaux
- Utiliser la version 3 pour la sécurité

SNMP: Architecture

- System Management Information
  - Système de nommage, définition et d'encodage des objets (RFC 2578)
- Management Information Base
  - Ensemble virtuel d'objets décrits et identifiés par des OID
  - Organisation des objets sous forme arborescente.
  - Différentes MIB dans des modules
- SNMP
  - Protocole pour lire et écrire des informations
  - Messages échangées en UDP entre client (manager) et serveur (agent)



SNMP: MIB



- 1.3.6.1.2.1 - SNMP MIB-2

- 1.3.6.1.2.1.1.1 - sysDescr
- 1.3.6.1.2.1.1.2 - sysObjectID
- 1.3.6.1.2.1.1.3 - sysUpTime
- 1.3.6.1.2.1.1.4 - sysContact
- 1.3.6.1.2.1.1.5 - sysName
- 1.3.6.1.2.1.1.6 - sysLocation
- 1.3.6.1.2.1.1.7 - sysServices

SNMP: Commandes

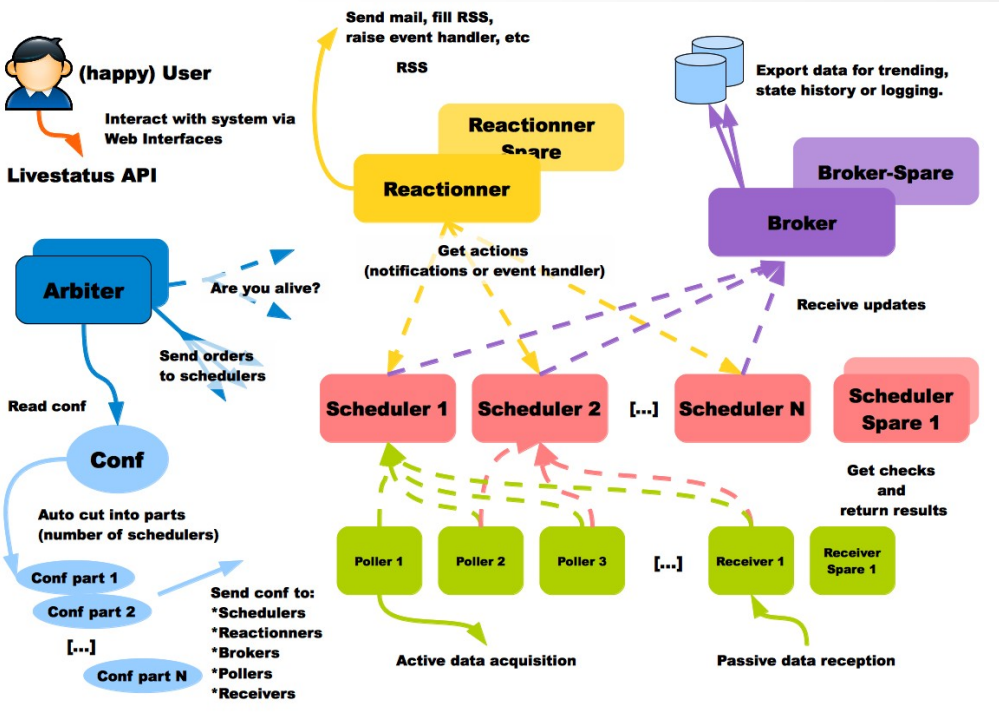
- Snmpget : récupère la valeur d'un OID feuille
  - Snmpget -v 1|2|3 172.16.252.2 .1.3.6.1.2.1.1.1
  - Snmpget -v 1|2|3 172.16.252.2 .iso.org.dod.internet.mgmt.mib-2.system.sysDescr
- Snmpset : assigne une valeur à un OID
- Snmpwalk: récupère toutes les valeurs d'un sous-arbre
- Snmpbulkget : récupère un ensemble de valeurs en une seule requête
- Snmptranslate : transforme une MIB en arbre (texte)
- Voir le site :
  - http://net-snmp.sourceforge.net/

Shinken

- Architecture modulaire et distribuée
- 6 composants intégrant des possibilités de modules
- Communications RPC sur middleware *Pyro* Python Remote Object
- Performant et extensible
- Compatible Nagios
  - Fichiers de configuration
  - Plugins de check
  - API Nagios (API LiveStatus), compatible Thruk.
- Nouvelles fonctionnalités
  - Corrélations
  - Vues Métiers

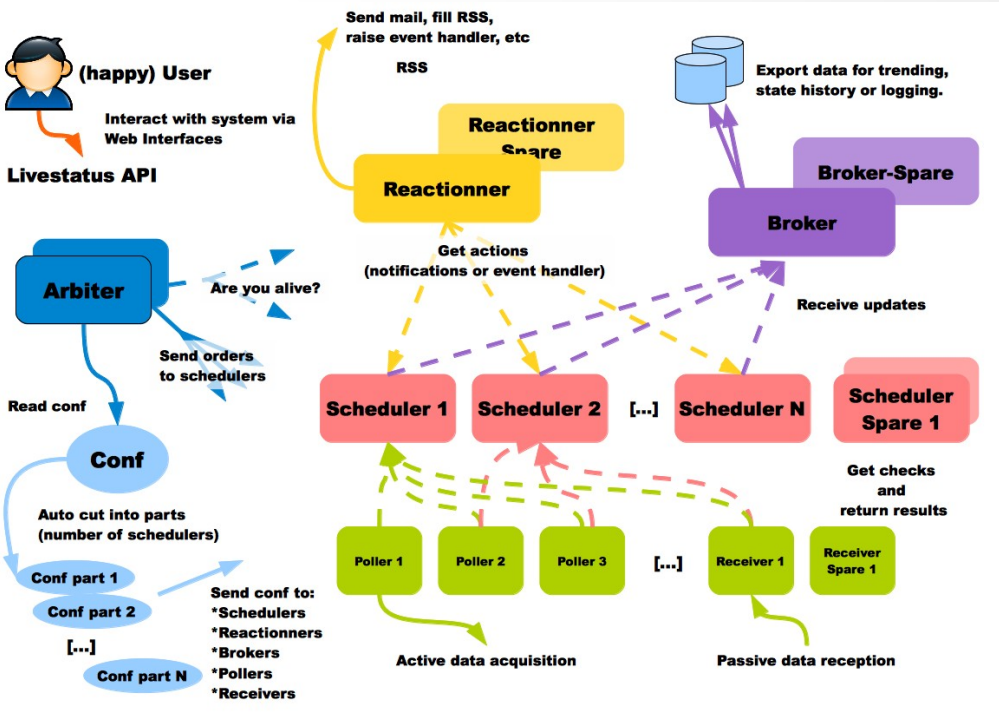
Shinken: Arbiter

- Lecture des fichiers de configuration, **répartition** et propagation aux autres composants
- Réception et gestion des commandes extérieures
- Redondance en mode actif/passif
- Surveillance des autres démons



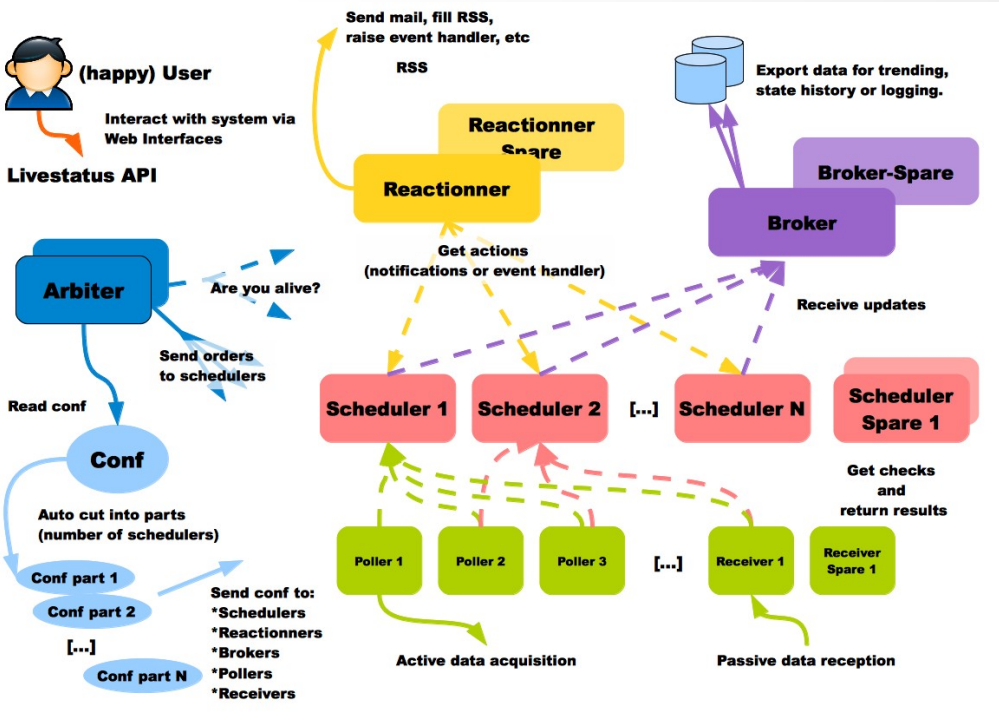
Shinken: Scheduler

- Ordonnancement des checks actifs
- Soumission aux Pollers
- Redondance en mode actif/actif
- Répartition de charge



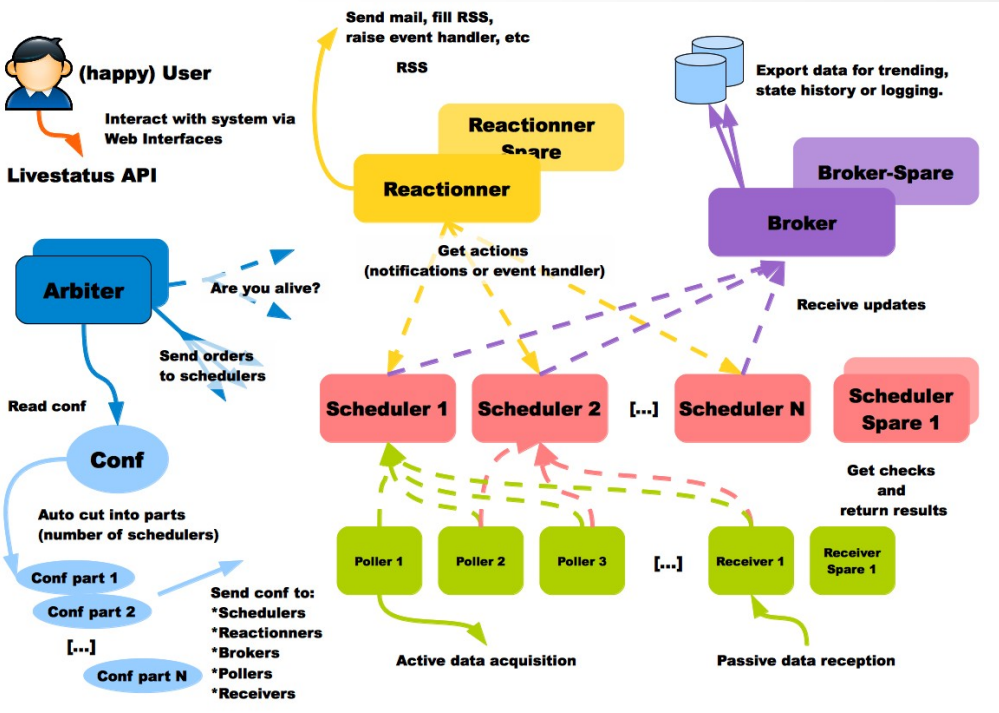
Shinken: Poller

- Exécution des checks actifs
- Envoi des résultats aux schedulers
- Redondance en mode actif/actif
- Répartition de charge



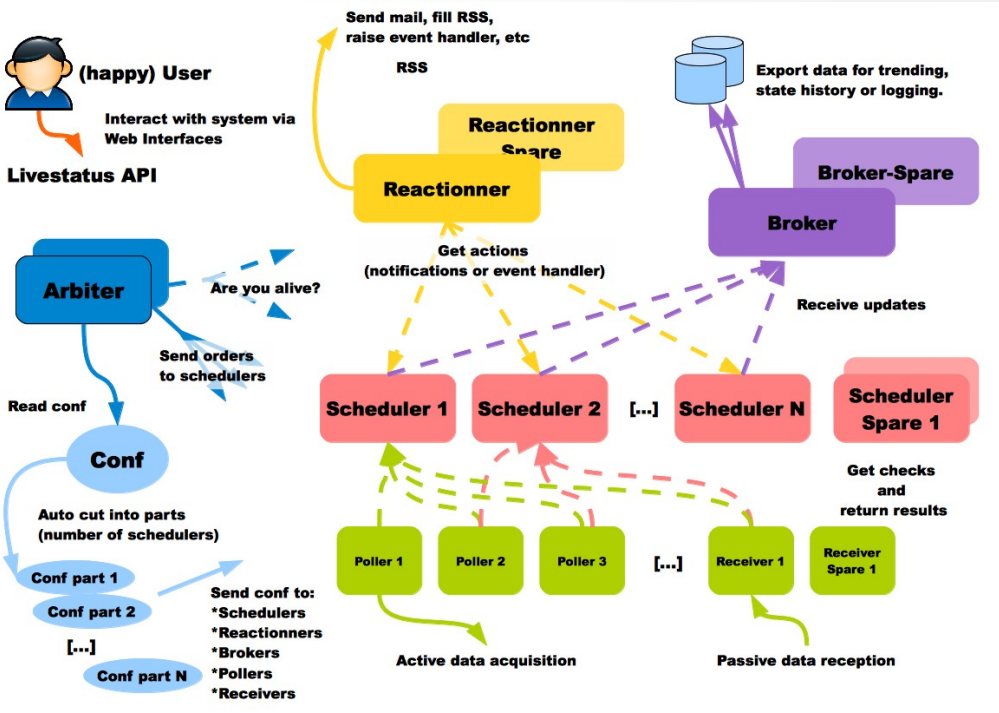
Shinken: Receiver

- Réception des résultats des checks passifs
- Envoi des résultats aux schedulers
- Redondance en mode actif/actif
- Répartition de charge



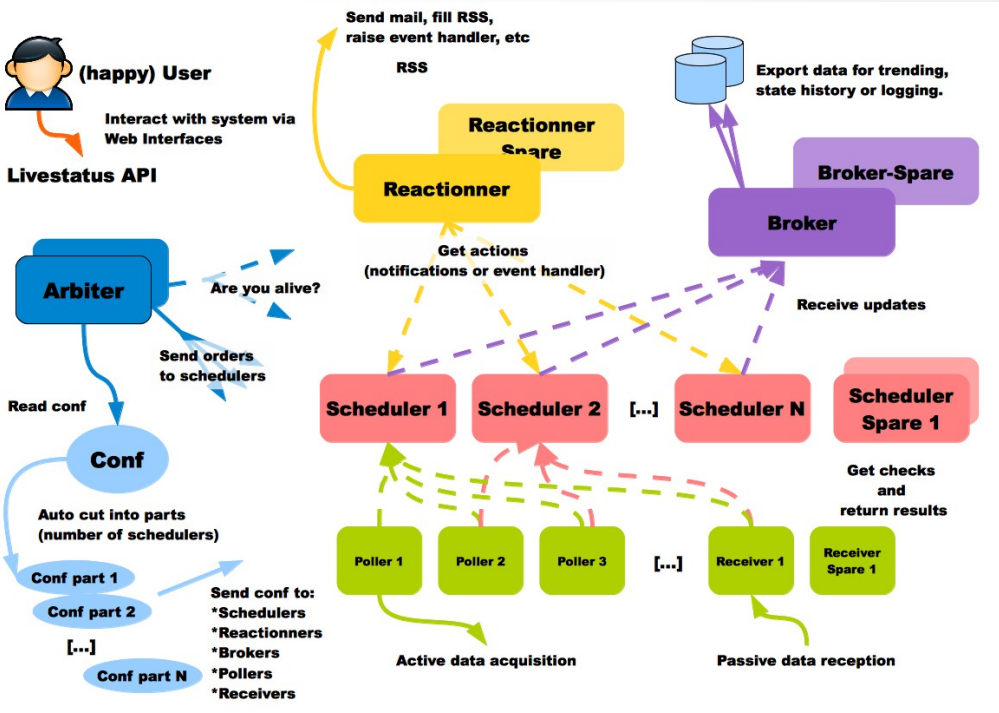
Shinken: Reactionners

- Envoi des alertes aux administrateurs
- Exécution des gestionnaires d'événements
- Redondance en mode actif/passif

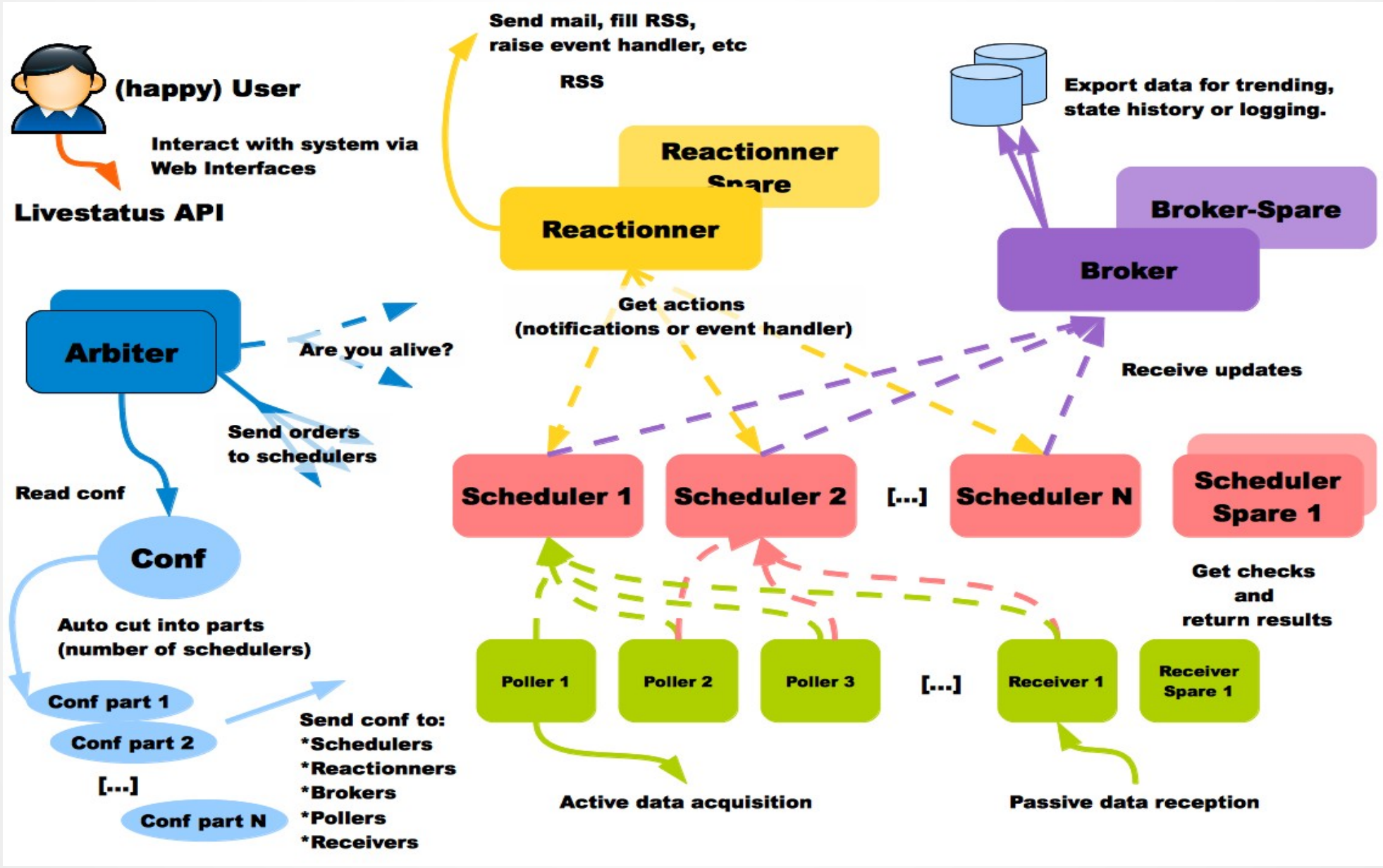


Shinken: Brokers

- Export des données d'état des schedulers
  - API Web LiveStatus
- Historisation
  - Base SQL
- Redondance en mode actif/passif
- Répartition de charge



Shinken : Architecture



Shinken: supervision active

- Méthode active
  - Lancement d'une commande pour récupérer l'information
  - Convient pour des vérifications de de services de courte durée
  - Via ssh, Nagios-NRPE, snmp-get
  - Avec ou sans rebond
  - Configuration centralisée simple

Shinken: supervision passive

- Méthode passive
  - Envoi de l'information par l'élément surveillé
  - Convient pour
    - des services "longs"
      - Ex : surveillance des sauvegardes de systèmes de fichiers
    - des services asynchrones
      - Ex : Alerte, trap snmp
    - Des services inaccessibles depuis le serveur shinken
      - Machine derrière un pare-feu
  - Gestion de configuration décentralisée plus complexe
  - Envoi des informations par des modules spécifiques
    - NSCA, TSCA, Shinken WebService

Shinken: Commandes

- Objet command
  - Définit une commande de vérification, notification, etc..
  - **command_name** définit le nom utilisé dans la configuration
  - **command_line** définit la commande linux et ses arguments
  - Peut utiliser des $MACROS$
  - http://shinken.readthedocs.io/en/latest/08_configobjects/command.html

```
define command{
    command_name    check_pop
    command_line    /var/lib/shinken/libexec/check_pop -H $HOSTADDRESS$
}
```

Shinken : Macros

- Permettent d'utiliser des valeurs définies pour les hosts, les services et autres objets.
- http://shinken.readthedocs.io/en/latest/05_thebasics/macros.html

```
define host{
  host_name    linuxbox
  address    192.168.1.2
  check_command    check_ping
  ...
}

define command{
  command_name    check_ping
  command_line    /var/lib/shinken/libexec/check_ping -H $HOSTADDRESS$ -w 100.0,90% -c 200.0,60%
}
```

```
/var/lib/shinken/libexec/check_ping -H 192.168.1.2 -w 100.0,90% -c 200.0,60%
```

```
define service{
  host_name    linuxbox
  service_description    PING
  check_command    check_ping!200.0,80%!400.0,40%
  ...
}
```

```
define command{
  command_name    check_ping
  command_line    /var/lib/shinken/libexec/check_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$
}
```

```
/var/lib/shinken/libexec/check_ping -H 192.168.1.2 -w 200.0,40% -c 400.0,80%
```

Shinken: Commandes de notification

- Commandes exécutées pour avertir les personnes
- Peut utiliser différents moyens :
  - Mail, sms, beeper

```
[root@vm-pg-shinken-1 ~]# tail -12 /etc/shinken/commands/commands.cfg
#### Now notification commands
define command{
        command_name                    notify-host-by-email
        command_line                    /usr/bin/printf "%b" "Shinken Notifi
cation\n\nType:$NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATE$\nAdd
ress: $HOSTADDRESS$\nInfo: $HOSTOUTPUT$\nDate/Time: $DATE$" | /bin/mail -s "
Host $HOSTSTATE$ alert for $HOSTNAME$!" $CONTACTEMAIL$
}


define command{
        command_name                    notify-service-by-email
        command_line                    /usr/bin/printf "%b" "Shinken Notifi
cation\n\nNotification Type: $NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nH
ost: $HOSTALIAS$\nAddress: $HOSTADDRESS$\nState: $SERVICESTATE$\n\nDate/Time
: $DATE$ Additional Info : $SERVICEOUTPUT$" | /bin/mail -s "** $NOTIFICATION
TYPE$ alert - $HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ **" $CONTACTEMAIL$
}
```

Shinken: Période de temps

- Permet de définir quand
  - les vérifications ou les notifications doivent être planifiées
  - http://shinken.readthedocs.io/en/latest/08_configobjects/timeperiod.html
  -

```
define timeperiod{
    timeperiod_name         nonworkhours
    alias                   Non-Work Hours
    sunday                  00:00-24:00             ; Every Sunday of every week
    monday                  00:00-09:00,17:00-24:00 ; Every Monday of every week
    tuesday                 00:00-09:00,17:00-24:00 ; Every Tuesday of every week
    wednesday               00:00-09:00,17:00-24:00 ; Every Wednesday of every week
    thursday                00:00-09:00,17:00-24:00 ; Every Thursday of every week
    friday                  00:00-09:00,17:00-24:00 ; Every Friday of every week
    saturday                00:00-24:00             ; Every Saturday of every week
}

define timeperiod{
    timeperiod_name         misc-single-days
    alias                   Misc Single Days
    1999-01-28              00:00-24:00             ; January 28th, 1999
    monday 3                00:00-24:00             ; 3rd Monday of every month
    day 2                   00:00-24:00             ; 2nd day of every month
    february 10             00:00-24:00             ; February 10th of every year
    february -1             00:00-24:00             ; Last day in February of every year
    friday -2               00:00-24:00             ; 2nd to last Friday of every month
    thursday -1 november    00:00-24:00             ; Last Thursday in November of every year
}

define timeperiod{
    timeperiod_name                 misc-date-ranges
    alias                           Misc Date Ranges
    2007-01-01 - 2008-02-01         00:00-24:00     ; January 1st, 2007 to February 1st, 2008
    monday 3 - thursday 4           00:00-24:00     ; 3rd Monday to 4th Thursday of every month
    day 1 - 15                      00:00-24:00     ; 1st to 15th day of every month
    day 20 - -1                     00:00-24:00     ; 20th to the last day of every month
    july 10 - 15                    00:00-24:00     ; July 10th to July 15th of every year
    april 10 - may 15               00:00-24:00     ; April 10th to May 15th of every year
    tuesday 1 april - friday 2 may  00:00-24:00     ; 1st Tuesday in April to 2nd Friday in May
}
```

Shinken: Hosts

- Représente un équipement à surveiller:
  - Un nœud, un commutateur réseau, un équipement
  - État : UP, DOWN, UNREACHABLE
  - http://shinken.readthedocs.io/en/latest/08_configobjects/host.html

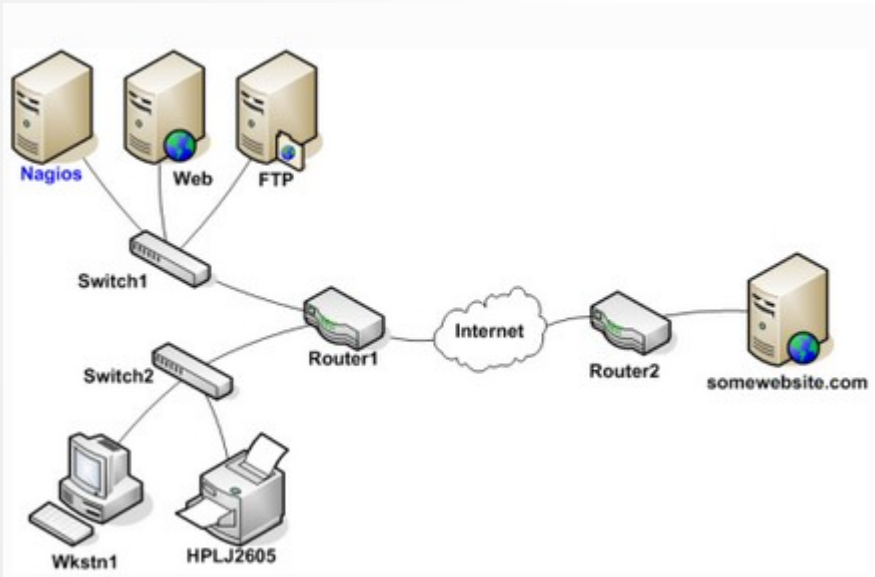
```
define host{
       host_name                       bogus-router
       alias                           Bogus Router #1
       address                         192.168.1.254
       parents                         server-backbone
       check_command                   check-host-alive
       check_interval                  5
       retry_interval                  1
       max_check_attempts              5
       check_period                    24x7
       process_perf_data               0
       retain_nonstatus_information    0
       contact_groups                  router-admins
       notification_interval           30
       notification_period             24x7
       notification_options            d,u,r
       realm                           Europe
       poller_tag                      DMZ
       icon_set                        server
       }
```

Shinken : host options

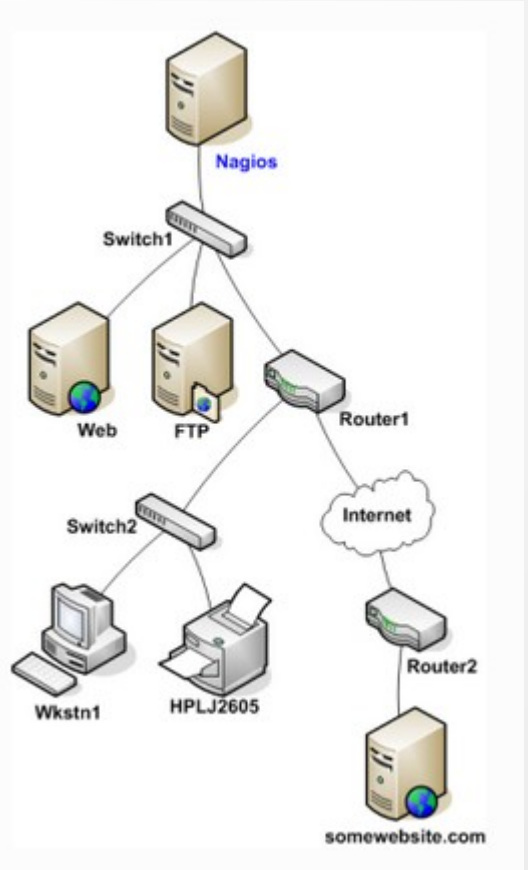
- check_command : donne le nom de la commande utilisé pour vérifier si le host est UP ou DOWN (ping)
- host_check_timeout : temps max d'exécution pour la commande précédente
- max_check_attempts : nombre de max de tentatives
  - 1=> notification immédiate
- check_interval : périodicité de la commande de vérification.
  - N fois une certaine unité de temps interval_length (en général 1 minute)
- retry_interval : période d'attente avant de une nouvelle vérification

Shinken: Hosts hierarchie

- Description de dépendances
  - http://shinken.readthedocs.io/en/latest/05_thebasics/networkreachability.html#thebasics-networkreachability



Définir une relation de parenté



Shinken: Contacts

- Représente une personne qui peut être avertie
  - http://shinken.readthedocs.io/en/latest/08_configobjects/contact.html

```
define contact{
    contact_name                    jdoe
    alias                           John Doe
    host_notifications_enabled      1
    service_notifications_enabled   1
    service_notification_period     24x7
    host_notification_period        24x7
    service_notification_options    w,u,c,r
    host_notification_options       d,u,r
    service_notification_commands   notify-service-by-email
    host_notification_commands      notify-host-by-email
    email                           jdoe@localhost.localdomain
    pager                           555-5555@pagergateway.localhost.localdomain
    address1                        xxxxx.xyyy@icq.com
    address2                        555-555-5555
    can_submit_commands             1
}
```

Shinken : options de notifications

- host_notification_options : définit les états d'un host pour lesquels un contact sera notifié
  - d = notifie si le host est DOWN
  - u = notifie si le host est UNREACHABLE
  - r = notifie si le host redevient UP (recovery)
  - f = notifie quand le host commence et arrête le flapping,
  - s = notifie quand le host entre et sort dans une période de maintenance (scheduled downtime).

- service_notification_options : définit les états d'un service pour lesquels un contact sera notifié
  - w = notifie si le service est WARNING
  - u = notifie si le service est UNKNOWN
  - c = notifie si le service est CRITICAL
  - r = notifie si le service est redevient OK (recovery)
  - f = notifie when the service commence et arrête le flapping,
  - n = (none) : le contact ne recevra aucun type of service notifications.

Shinken: ContactsGroup

- Représente un groupe de personnes qui peuvent être averties
  - http://shinken.readthedocs.io/en/latest/08_configobjects/contactgroup.html

```
define contactgroup{
    contactgroup_name       novell-admins
    alias                   Novell Administrators
    members                 jdoe,rtobert,tzach
}
```

Shinken: Services

- Un service est un point de supervision
  - Service système s'executant sur un host :
    - Serveur ntp, rsyslogd, serveur http, database
  - Une métrique à surveiller :
    - Charge CPU, Réseau, Erreurs Interconnection, Nombre de licences, Nombre de nœuds de login UP
  - Plusieurs services sur un seul host
  - Etat : OK, WARNING, CRITICAL, UNKNOWN
  - http://shinken.readthedocs.io/en/latest/08_configobjects/service.html

```
define service{
       host_name               linux-server
       service_description     check-disk-sdal
       check_command           check-disk!/dev/sdal
       max_check_attempts      5
       check_interval          5
       retry_interval          3
       check_period            24x7
       notification_interval   30
       notification_period     24x7
       notification_options    w,c,r
       contact_groups          linux-admins
       poller_tag              DMZ
       icon_set                server
       }
```

Shinken : Templates

- Beaucoup d'éléments de configuration identiques
- Mutualisation de configuration
- 2 mots clés :
  - **Register** 0
  - **Use** <modele>

```
[root@vm-pg-shinken-1 ~]# cat /tmp/templates.cfg
define host{
        name generic-host
        alias generic-host
        max_check_attempts 2
        check_interval 5
        active_checks_enabled 1
        check_period 24x7
        notification_options d,u,r,f
}

define host{
        host_name sshgw
        use generic-host
        address         172.67.1.3
        parents SwitchBB1,SwitchLA2
        use generic-host
}
```

```
# Generic service definition template
# This is NOT a real service, just a template!
define service{
        name                            generic-service
        active_checks_enabled           1
        passive_checks_enabled          1
        check_period                    24x7
        max_check_attempts              3
        check_interval                  5
        retry_interval                  1
        contact_groups                  admins
        notifications_options           w,u,c,r
        notification_interval           0
        notification_period             24x7
        register                        0
}

#This one is a real one
define service{
        name                            local-service
        use                             generic-service
        max_check_attempts              1
        register                        0
}
```

Shinken : états

- Etat courant d'un service ou d'un hote
  - Son état : UP, DOWN, UNKNOWN,
  - Le type d'état : SOFT ou HARD
- Le type SOFT ou HARD détermine
  - Quand le gestionnaire d’événements est exécuté
  - Quand les notifications sont envoyées

Shinken : État Soft

- État Soft : ~ Soft Error :
  - état KO et Nb de contrôle < **max_check_attempts**.
    - L'erreur n'a pas encore été confirmée !
  - Ou état rétabli OK après passage en soft error

- Le passage à cet état entraîne
  - L'enregistrement de cet état si log_service_retries = 1
  - L’exécution d' un gestionnaire d’événements (event handler)
    - $SERVICESTATETYPE$ = SOFT

- Permet des actions correctives

Shinken : État Hard

- État Hard :
  - État initial
  - État pas OK (UP) et nombre de contrôle = **max_check_attempts**.
  - Transition d'un état d'erreur à un autre état d'erreur Warn → Crit
  - Quand un service redevient OK à partir d'un état hard
    - Hard recovery

- Le passage à cet état entraîne
  - L'enregistrement de cet état HARD
  - L’exécution d' un event handler $SERVICESTATETYPE$ = HARD
  - **Notification** du contact.

Shinken : Sondes

- Voir les sites :
  - http://www.shinken.io/
  - https://exchange.nagios.org/
- Moyens de lancement :
  - Linux/ssh surveillance d'un système linux sans installation de sonde sur le nœud
  - NRPE : Nagios Remote Plugin Executor
  - SNMP
- Code de retour ($?) sonde service :
  - **OK (0)**, **WARNING (1)**, **CRITICAL (2)**, **UNKNOWN (3)**
- Code de retour ($?) sonde host :
  - **UP (0)**, **DOWN (1), DOWN (2), DOWN (3)**
- La sonde peut donner quelques information sur son stdout :
  - $SERVICEOUTPUT$, $LONGSERVICEOUTPUT$,

Shinken : Écriture d'une sonde

- Code de retour ($?)
  - OK (0) Le plugin a pu tester le service et celui-ci fonctionne.
  - Warning (1) Le plugin a pu tester le service mais celui-ci semble au-dessus d'un seuil warning ou ne semble pas fonctionner correctement
  - Critical (2) Le plugin a détecté que le service semble au dessus d'un seuil critique ou ne tourne pas du tout.
  - Unknown (3) Des paramètres invalides ont été fournis au plugin ou une erreur interne de bas niveau (fork/socket) s'est produit pendant l'exécution Des erreurs de plus haut niveau indépendantes du plugin ne doivent pas être reportées comme UNKNOWN.

Shinken : Écriture d'une sonde

- Conventions pour les options
  - V version (--version)
  - -h help (--help)
  - -t timeout (--timeout)
  - -w warning threshold (--warning)
  - -c critical threshold (--critical)
  - -H hostname (--hostname)
  - -v verbose (--verbose)

Shinken : Écriture d'une sonde

- Conventions pour la sortie :
  - Pas de messages sur stderr
  - Toujours un message court sur stdout :
    - Quel service, OK, KO, une info
    - Peut être envoyé par sms ou sur un pager.
  - -v pour augmenter la volubilité
    - 0 : Par défaut, stdout minimal
    - 1 : Informations supplémentaires
    - 2 : Informations de debug (ex commande utilisée)
    - 3 : Beaucoup de détails pour diagnostiquer un problème

Shinken : Écriture d'une sonde

- Seuils warning et critique :
  - Permettent d'associer un état à une valeur
  - Les valeurs expriment des unités, par forcément des %
  - Espace libre sur un système de fichiers
  - Nombre d'erreurs
  - Nombre de paquets perdus
  - Occupation mémoire

Shinken : Écriture d'une sonde

- Règles d'écriture :
  - Utiliser les paths complets pour les commandes ou spécifier PATH=
  - Être économes de ressources
    - Pas de fichiers temporaires
    - Pas de résolutions de noms
    - Pas de résolutions d'adresses
    - Pas d'appel à des services externes
  - Pas de stress à l’exécution (scalable)
  - Une sonde doit s’exécuter très rapidement < 10 sec

Shinken : Références

- https://nagios-plugins.org/doc/guidelines.html#DEVREQUIREMENTS
- https://shinken.readthedocs.io/en/latest/