Le logiciel de supervision Nagios

Comment choisir un logiciel de supervision? Exemple de critères de sélection. Pourquoi choisir Nagios ?

- Nagios logiciel de supervision :
  - open source, stable (à justifier), performant (à justifier)
  - ayant une forte communauté
  - référence dans le monde de la supervision Open Source.

- Nagios : avant tout un ordonnanceur
  - lance à intervalles réguliers des plugins (sondes)
  - récupère les informations renvoyées par les sondes
  - Analyse les informations recueillies :
    - Si elles ne dépassent pas des limites fixées par l'administrateur → OK
    - Si dépassent les limites → envoie de notifications (mail, SMS ...)

- La modularité/flexibilité de Nagios grâce aux plugins :
  - Plugins développés par la « communauté ». Leur conception est très simple (souvent de simples scripts Bash, Perl, Python).
  - Des plugins sont également définis pour avertir ou réagir (envoie d'email, de sms, actions à entreprendre).

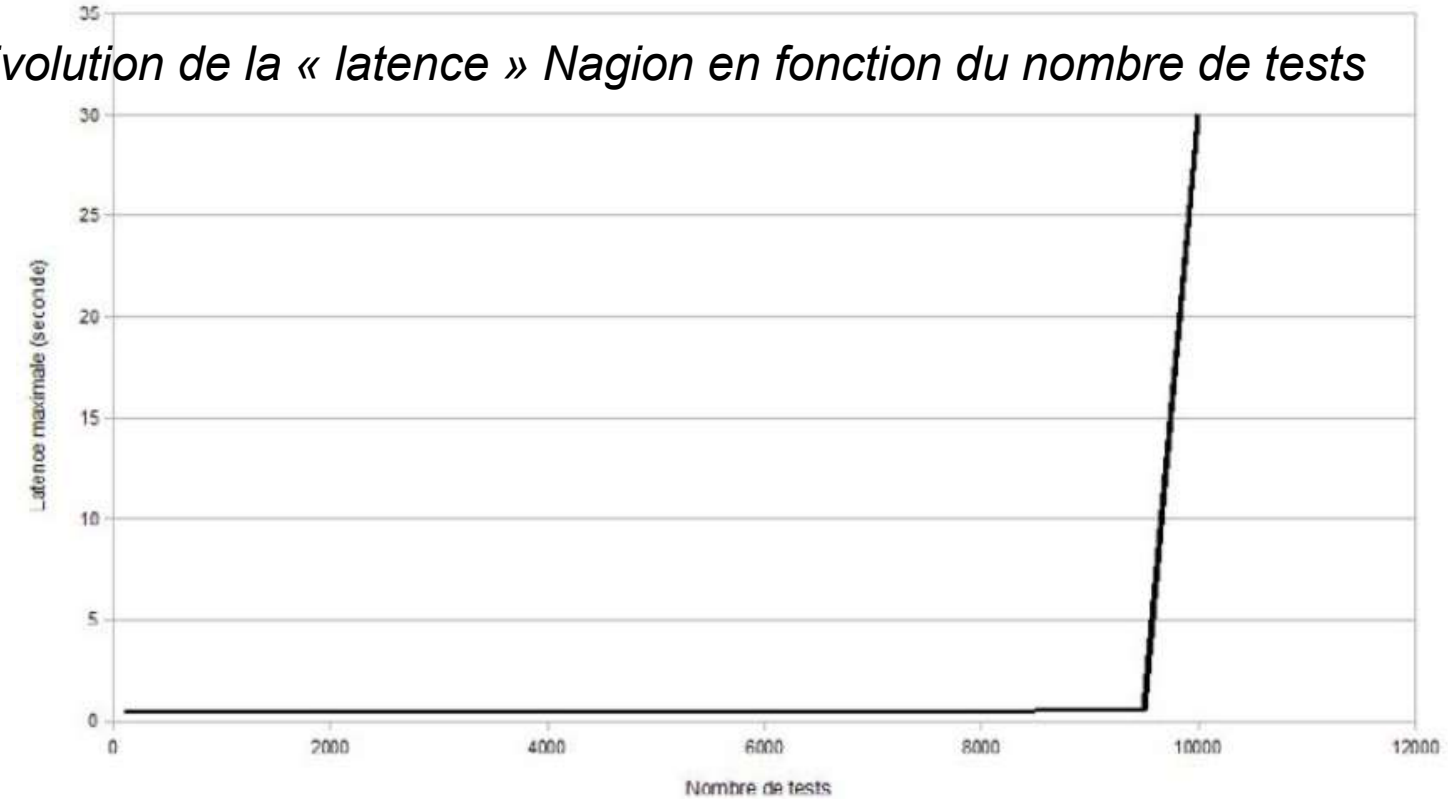
- Capacités à gérer un parc important :
  - Performances (ressource mémoire et processeur d'un tests) → Combien de tests par seconde ?
  - Gestion des configurations via des modèles
  - Analyse et synthèse des alertes
  - Haute disponibilité ?
  - Répartition de charge (scalabilité) : deux aspects essentiels pour des très gros SI à superviser. Pas des points forts de Nagios.

Nagios – Capacité à gérer un parc important

Capacité à gérer un parc important de machines

- Les performances
  - Supervision = lancer régulièrement un nombre élevé de vérifications sur des hôtes distants.
  - Ordonnancement différent en fonction de l'élément surveillé.
    - Supervision de la charge CPU différente de celle des espaces disques. CPU varie très vite (un serveur qui ne fait pas grand chose à 8H50 peut être surchargé à 9H). La probabilité est beaucoup plus faible de voir un disque ayant 2To d'espace libre se remplir en moins de 5 minutes.
  - Lancer un test a un coût. La charge d'un test est très faible, celle de plusieurs milliers élevée.
  - Nagios doit pouvoir lancer un grand nombre de tests par minute. S'il n'y arrive pas, il va prendre du retard : la latence (jargon Nagios).

Evolution de la « latence » Nagion en fonction du nombre de tests



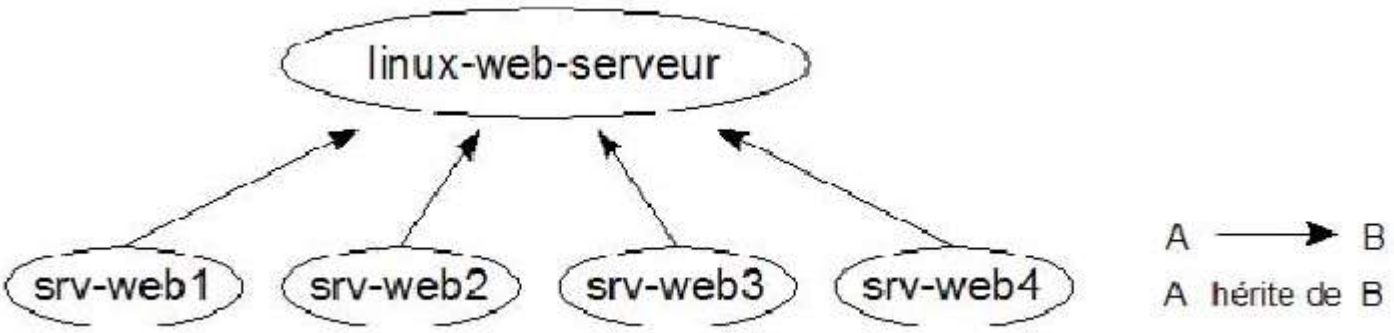
Source : Nagios 3 pour la supervision et la métrologie Déploiement, configuration et optimisation

Nagios – Capacité à gérer un parc important

- Gestion de la configuration
  - Plus on a de points à surveiller, plus la configuration devient lourde. Risque si elle devient trop complexe à gérer : laissée de côté.
  - Exemple d'information pour superviser une machine :

```
define host {
host_name    srv-web1
alias    Serveur web 1
hostgroups linuxservers
address  192.168.0.1
check_command    check-host-alive
check_interval   5
retry_interval   1
max_check_attempts   3
check_period 24x7
contact_groups   web-admins
notification_interval30
notification_period  24x7
notification_options d,u,r ; Etat DOWN, UNREACHABLE, revient
UP
}
```

  - Pour faciliter la configuration, Nagios :
    - permet de définir des modèles et des groupes de machines
    - utilise un principe d'héritage pour la gestion des configuration.



Nagios – Capacité à gérer un parc important

Gestion de la configuration (suite)

Exemple :

```
define host{
  name  linux-web-server
  contacts_group  admin-linux,admin-apache
  [...]
  register  0 ; définit un modèle
}

define host{
  use   linux-web-server
  host_name  srv-web1
  alias  srv-web1
  address  srv-web1.mydomain.com
}

define host{
  use  linux-web-server
  host_name  srv-web2
  [...]
}
```

Toutes les lignes définies dans linux-web-server n'ont pas besoin d'être réécrites ici

- Exemple 2 : Définition de
  - un modèle pour toute les machines serveurs
  - puis un modèle par type d'OS qui hérite du modèle serveur
  - puis un modèle pour les machines de type serveur-web qui hérite des propriétés du modèle serveurs linux ou serveur windows
  - ….

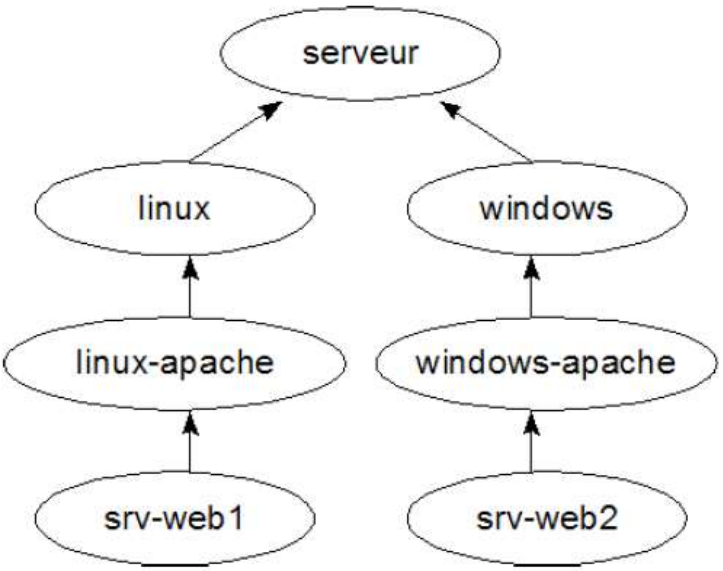
Nagios – Capacité à gérer un parc important

```
define host{
  name  generic-server
  contacts_group  admin-apache
  [...]
  register  0 ; défini un modèle
}
define host{
  use  generic-server
  name  linux-web-server
  check_command  check_tcp!22
  register  0
}

define host{
  use  generic-server
  name  windows-web-server
  check_command  check_tcp!339
  register  0
}

define host{
  use  linux-web-server
  host_name  srv-web1
  alias  srv-web1
  address  srv-web1.mydomain.com
}

define host{
  use  windows-web-server
  host_name  srv-web2
  [...]
}
```

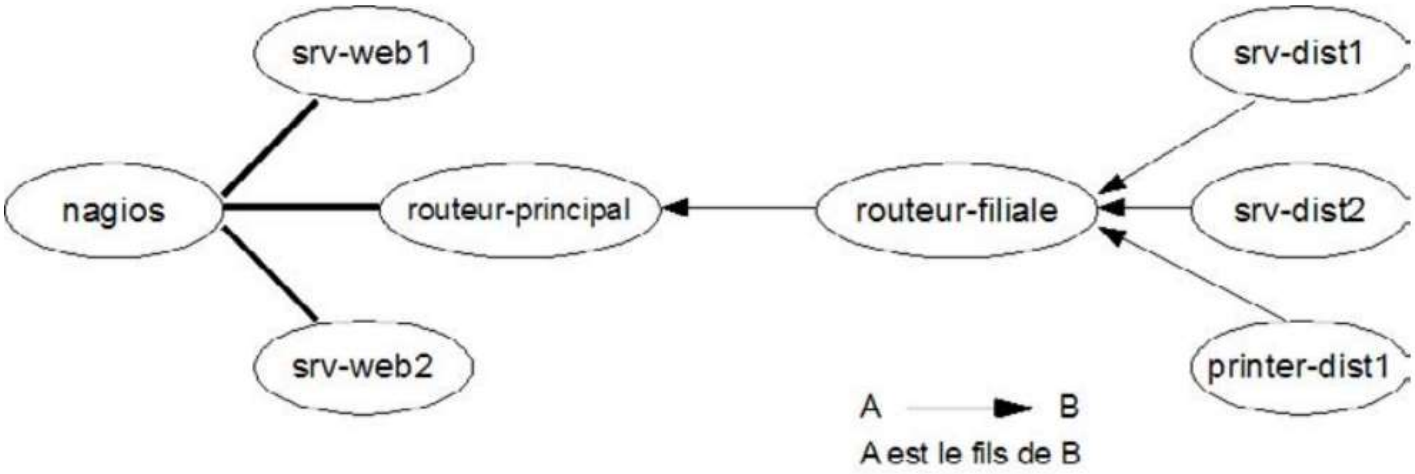


Remarque possibilité d'avoir de l'héritage multiple.

Nagios – Capacité à gérer un parc important

Analyse et Synthèse des alertes

- SI actuels vastes
- Des éléments inter-dépendants. Si l'un d'eux rencontre des problèmes, ils se répercutent sur d'autres éléments.
- Un simple dysfonctionnement ==> un grand nombre d'alertes.
- Difficultés à trouver, parmi toutes les alertes, la **cause initiale du problème**.
- Nagios gère ces cas grâce aux relations de dépendances. Ces relations peuvent être :
  - physiques (exemple des liens réseaux)
  - virtuelles (comme c'est le cas entre une application est une base de données).

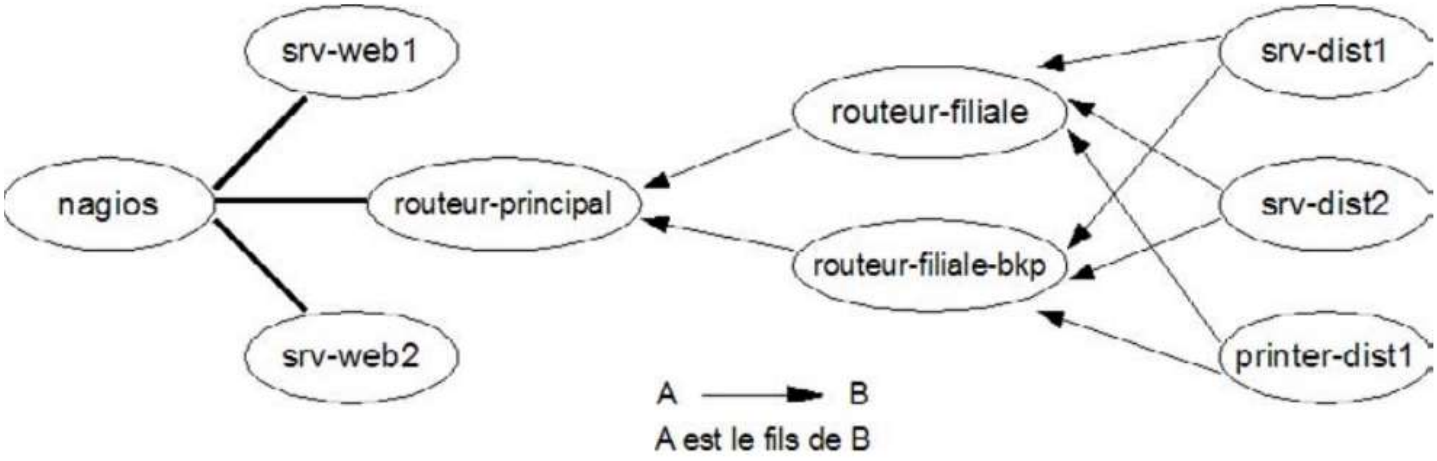


Perte du lien, l'ensemble des éléments de la filiale vont remonter une erreur à Nagios ==> beaucoup trop d'alertes.

Un élément peut avoir un ou plusieurs pères auquel il est connecté. Lorsqu'il rencontre un problème sur un noeud, Nagios remonte son arbre de parenté en lançant des test de disponibilité. Le dernier élément ne répondant pas est la cause du problème.
Cet élément passe en état ***DOWN***, ses fils sont en état ***UNREACHABLE***.

Nagios – Capacité à gérer un parc important

- Cas de relations à parents multiples, il faut que tous les parents soient non disponibles pour que l'enfant soit déclaré non joignable.




```
define host{
  host_name  srv-dist1
  parents routeur-filiale,routeur-filiale-bkp
}

define host{
  host_name  routeur-filiale
  parents routeur-princ
}

define host{
  host_name  routeur-filiale-bkp
  parents routeur-princ
}
```

Fichiers de configuration de nagios

- Définition d'un hôte :

```
define host {
host_name    srv-web1
alias    Serveur web 1
address  192.168.0.1
check_command    check-host-alive
check_interval   5
retry_interval   1
max_check_attempts   3
check_period 24x7
contact_groups   web-admins
notification_interval 30
notification_period  24x7
notification_options d,u,r ; DOWN, UNREACHABLE, revient UP
}
```

Quelle commande permet selon vous de tester que l'hôte est toujours en vie ? Que fait réellement cette commande ?

A quoi correspondent selon vous les lignes commençant par ***notification*** *?*

- Définition d'un service.
  - Services = éléments supervisés sur les hôtes. Ex : fonctionnement d'une application particulière.
  - Rmq : Un service est un point de supervision.Une vérification CPU est également un service.

```
define service{
host_name srv-web1
service_description Http
check_command check_tcp!80
max_check_attempts 2
[...]
```

Les Plug-in – Les commandes

- Nagios lance des commandes de vérification et utilise les codes de retour. Code retour : valeur renvoyée à celui qui a lancé le programme, et qui indique si celui-ci a fonctionné correctement ou s'il a rencontré un problème.

- Codes retour « classiques » :
  - 0 --> la commande s'est exécutée avec succès
  - 1--> la commande a rencontré un problème mineur
  - 2 --> la commande a rencontré un problème majeur

- Pour Nagios la signification des codes de retour diffère pour un hôte ou un service :
  - Pour les hôtes :
    - 0 : UP
    - 1 : UP/DOWN
    - 2 : DOWN
    - 3 : DOWN

Dans le cas des alertes DOWN, si le noeud se trouve derrière un élément réseau non disponible, il sera en état UNREACHABLE.

  - Pour les services :
    - 0 : OK
    - 1 : WARNING
    - 2 : CRITICAL
    - 3 : UNKNOWN

- Données de métrologie : remontées via la sortie standard. Pour les informations de métrologie, le plugin utilise le séparateur **|**.

```
DISK OK | /=56% /boot=50% /home=15%
```

Nagios Types d'état SOFT et HARD

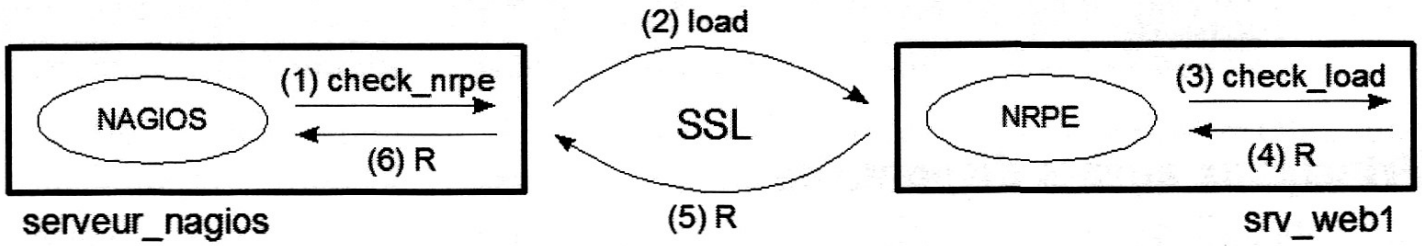
- Logique de supervision de Nagios : les **notifications** sont envoyées uniquement si les états des hôtes et des services ont été **validés**.

- Voir documentation et exercice : https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/statetypes.html

Nagios – Interrogation des systèmes distants

- Solution recommandée : installation d'un agent sur les serveurs supervisés.

- Agent Nagios sur les hôtes NRPE- Nagios Remote Process Executor.
  - NRPE agent à installer sur les machines distantes. Agent : logiciel qui réceptionne les demandes de Nagios, exécute un script récupère l'information et lui renvoient.
  - NRPE permet d'exécuter à distance des plugins Nagios classiques déposés par l'administrateur.




```
check_nrpe -H srv-web1 -c load
OK - Charge moyenne: 0.51, 0.28, 0.28|load1=0.510;1.000;2.000;0;
load5=0.280;1.000;2.000;0; load15=0.280;1.000;2.000;0;

echo $?
0
```

  - NRPE : solution la plus souvent utilisée.
  - Inconvénient pas disponible directement sous Windows.

- Environnement Microsoft Windows (voir Internet).

Nagios – Interrogation des systèmes distants

- Interrogation via SNMP (matériel réseau)
  - Plugin de base : **check_snmp** :

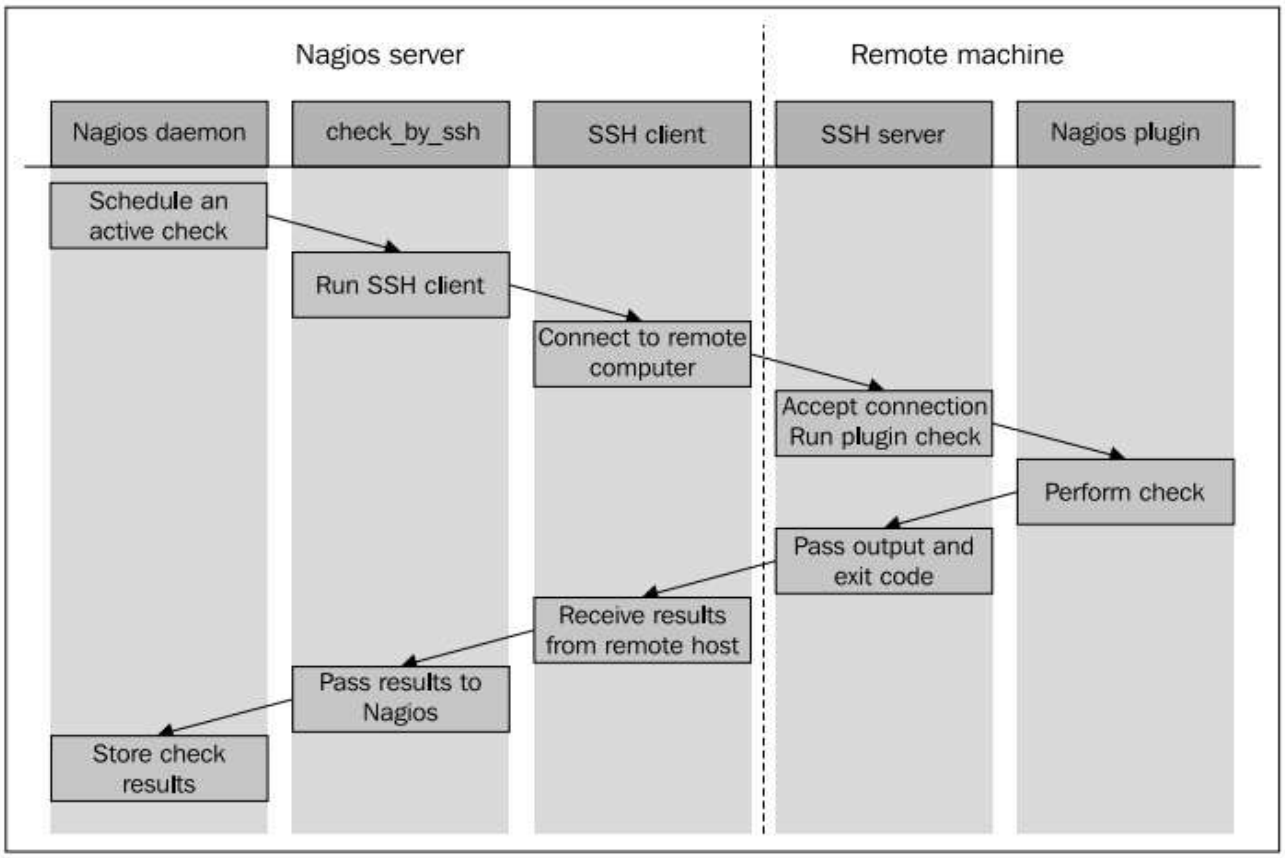
```
check_snmp -H <ip_address> -o <OID> [-w warn_range] [-c
crit_range] [-C community] [-s string] [-r regex] [-R
regexi] ...
```

  - Plugins supplémentaires exemple :

```
check_snmp_switch_cpu.pl -H 192.168.0.254 -C public -w 50 -c
60
SNMP_ENVIRONMENT OK : CpuUtil=10%|CpuUtil=10%
```

  - Parfois de simples vérifications via réseau ne suffisent pas. Il n'y a pas d'OID SNMP correspondant. Ex charge du service bind sur un serveur ?
  - De manière générale les admins sys utilisent rarement snmp.

- Interrogation via SSH + Plugin (si pas d'agent pour le système à superviser).



  - PB SSH : Une vérification consomme beaucoup de ressources.

Nagios – Solutions proches

Fork Centreon-Icinga-EyesOfNetwork-Shinken ...

- Objectifs de ces solutions :
  - Faciliter la configuration
  - Interfaces graphiques plus « performante »
  - Stockage des configurations en BDD et non pas en fichiers.
  - Affichage des données métrologiques
  - Scalability : possibilité de scinder le logiciel en différents composants

Solution ayant une forte croissances actuelles

- Zabbix (monitoring classique)
- Prometheus : plus orienté supervision système distribué conteneur/vm.
- Grafana : création de Dashboard pour visualiser les données. Générer des alarmes.
- Observabilité : OpenTelemetry, Dynatrace, Splunk,

La métrologie

- Métrologie : science de la mesure au sens le plus large. Mesure : opération qui consiste à donner une valeur à une observation.

- Métrologie réseau : mesure des performances du réseau :
  - Instrumenter le réseau et mesurer des caractéristiques du réseau
  - Analyser les mesures collectées
  - Repérer des comportements normaux/anormaux

- Supervision « classique » (monitoring) informe en temps réel de l'état des équipements, par contre ne permet pas de savoir si le réseau assure parfaitement le service pour lequel il a été conçu.

- Métrologie : notion de qualité de service / Supervision : avant tout disponibilité.

- Monitoring : lancer un nombre élevé de vérifications sur les hôtes distants. La métrologie doit conserver un nombre plus restreints d'éléments mais sur une durée assez longue.