

# Activité : Installation et test du serveur de mails Postfix

## Installation de Postfix

Postfix sera installé dans un premier temps avec un paramétrage basique. Lors de l'installation, choisissez **site internet** dans la configuration du paquet.

```
root@debianWheezy:# apt-get install postfix
```

Réponses à l'assistant d'installation :

- Type de configuration : Site internet ;
- Nom de courrier : Shinken ;

Le nom de courrier est le nom de votre serveur.

Modifiez le fichier **/etc/postfix/main.cf** pour **inhiber** la résolution par **DNS** qui est active par défaut, et forcer la prise en compte du fichier **/etc/hosts**.

- ajouter en fin de fichier la ligne suivante :

```
smtp_host_lookup = native
```

Ce paramètre force **postfix** à passer par le fichier **/etc/nsswitch.conf**, qui va lui donner comme méthode de résolution de noms file soit **/etc/hosts**.

Le fichier **/etc/nsswitch.conf** doit contenir la ligne suivante avec le mot **file** en premier, **dns** en second :

```
hosts: files dns
```

**Modifiez** le contenu du fichier **/etc/mailname** pour avoir le contenu suivant :

```
Shinken
```

**Redémarrez** le service Postfix :

```
root@Shinken:~# service postfix restart
```

## Création d'un utilisateur pour recevoir les notifications

Créez un utilisateur avec votre prénom ; dans la suite du document il s'agira de **roger**.

```
root@Shinken:~# adduser roger
```

## Test du serveur mail postfix et de l'utilisateur

- Envoyez un message avec la commande mail pour vérifier que ça fonctionne :

```
root@Shinken:~# mail -s "bonjour" roger
hello
j'essaie de t'envoyer un message
Cc:
Ctrl+d //pour sortir
```

**Attention** : Si le fichier **resolv.conf** ne définit pas un domaine de recherche par défaut, il faut mettre le nom complet, par exemple **roger@btssio.local**

Il suffit ensuite de se connecter avec roger (su - roger) et de vérifier qu'il a reçu le message avec la commande mail (pour revenir à root tapez exit).

```
roger@Shinken:~$ mail
```

```
Mail version 8.1.2 01/15/2001.  Type ? for help.
"/var/mail/roger": 1 message 1 new
>N 1 root@ctShinken    Tue Nov 22 22:49    14/396    Bonjour
&
```

- Tapez le numéro du message pour visualiser son contenu :

```
& 1
Message 1:
From root@Shinken  Tue Nov 22 22:49:52 2016
X-Original-To: roger
To: roger@ctShinken
Subject: Bonjour
Date: Tue, 22 Nov 2016 22:49:52 +0100 (CET)
From: root@ctShinken (root)

Hello
j'essaie de t'envoyer un message
& q // pour sortir
```

Cela fonctionne et Postfix est opérationnel pour ce que l'on veut faire. La suite consiste à configurer les notifications dans Shinken.

Revenez à root par la commande **exit**.

## Commandes de notification par défaut

Pour envoyer des mails, Shinken dispose de commandes définies dans `$ETC/notificationways` :

```
root@Shinken:~# cat /etc/shinken/notificationways/email.cfg
# This is how emails are sent, 24x7 way.
define notificationway {
    notificationway_name      email
    service_notification_period 24x7
    host_notification_period   24x7
    service_notification_options c,w,r
    host_notification_options   d,u,r,f,s
    service_notification_commands notify-service-by-email ; send service notifications via email
    host_notification_commands  notify-host-by-email      ; send host notifications via email
}
```

### Rappel :

- w (warning),
- c (critical),
- r (recovery),
- d (down),
- u (unknown).

On peut observer qu'une période de temps (24x7) est respectée pour envoyer les notifications. Cette période est définie dans **`$ETC/timeperiods/24x7.cfg`**

```
define timeperiod{
    timeperiod_name      24x7
    alias                Always
    sunday                00:00-24:00
    monday                00:00-24:00
    tuesday               00:00-24:00
    wednesday             00:00-24:00
    thursday              00:00-24:00
    friday                00:00-24:00
    saturday              00:00-24:00
}
```

La période 24x7 définit une plage de 24h sur 24 et de 7 jours sur 7.

Les commandes utilisées par le service de notification sont définies dans le répertoire **`$ETC/commands`**.

```
Fichier **notify-host-by-email.cfg** :
<code>
```

```
## Notify Host by Email
define command {
    command_name    notify-host-by-email
    command_line    /usr/bin/printf "%b" "Shinken Notification\n\nType:$NOTIFICATIONTYPE$\nHost:
$HOSTNAME$\nState: $HOSTSTATE$\nAddress: $HOSTADDRESS$\nInfo: $HOSTOUTPUT$\nDate/Time: $DATE$\n" |
/usr/bin/mail -s "Host $HOSTSTATE$ alert for $HOSTNAME$!" $CONTACTEMAIL$
}
```

Fichier **notify-service-by-email.cfg** :

```
## Notify Service by Email
define command {
    command_name    notify-service-by-email
    command_line    /usr/bin/printf "%b" "Shinken Notification\n\nNotification Type:
$NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost: $HOSTALIAS$\nAddress: $HOSTADDRESS$\nState:
$SERVICESTATE$\n\nDate/Time: $DATE$ Additional Info : $SERVICEOUTPUT$\n" | /usr/bin/mail -s "***
$NOTIFICATIONTYPE$ alert - $HOSTALIAS$/ $SERVICEDESC$ is $SERVICESTATE$ *" $CONTACTEMAIL$
}
```

L'envoi est réalisé en 2 étapes :

- il y a d'abord la **génération** du texte (par la commande **printf**),
- puis l'**envoi** par mail (par la commande **mail**).

Des macros (variables) sont utilisées, signalées par l'encadrement des \$.

Par exemple, la macro **\$HOSTNAME\$** sera remplacée par **ctShinken** dans le cas d'une alerte sur cette machine.

## Configuration du contact à alerter dans shinken

L'utilisateur **roger** créé précédemment existe pour Linux mais pas pour Shinken. Il faut **créer un contact** dans Shinken et l'**associer** à notre utilisateur.

Vous allez configurer ce contact dans le fichier **\$ETC/contacts/roger.cfg** qui faut créer. Attention : on rajoute un contact aux 2 contacts existants (admin et guest).

```
define contact{
    use                generic-contact
    contact_name       roger
    email              roger@shinken
}
```

Ce contact est identifié par le nom **roger**, il hérite de **generic-contact** qui définit des valeurs par défaut pour les paramètres de notification, notamment le droit de recevoir des notifications et la commande de notification utilisée.

**Rappel** : les modèles génériques sont définis dans le fichier **\$ETC/templates/generic-contacts.cfg**.

```
# Contact definition
# By default the contact will ask notification by mails
define contact{
    name                generic-contact
    register            0
    host_notifications_enabled 1
    service_notifications_enabled 1
    email              shinken@localhost
    can_submit_commands 1
    notificationways    email
}
```

## Configuration du déclenchement des alertes

La périodicité du déclenchement des notifications est définie de façon générale dans les modèles **generichost** et **generic-service**.  
**<code> # Generic service definition template - This is NOT a real service, just a template! define service { name generic-service ; The 'name' of this service template activechecksenabled 1 ; Active service checks are enabled passivechecksenabled 1 ; Passive service checks are enabled/accepted notificationenabled 1 ; Service notifications are enabled notificationinterval 10 notificationperiod 24x7 eventhandlerenabled 0 ; Service event handler is enabled**

**flapdetectionenabled 1 ; Flap detection is enabled processperfd data 1 ; Process performance data is volatile 0 ; The service is not volatile checkperiod 24x7 ; The service can be checked at any time of the day maxcheckattempts 3 ; Re-check the service up to 3 times in order to determine its final (hard) state checkinterval 5 ; Check the service every 5 minutes under normal conditions retryinterval 2 ; Re-check the service every two minutes until a hard state can be determined notificationoptions w,u,c,r ; Send notifications about warning, unknown, critical, and recovery events contactgroups admins,users stalkingoptions o,w,u,c register 0 ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL SERVICE, JUST A TEMPLATE } </code>** Ce fichier définit la façon dont Shinken vérifie l'état des objets supervisés : \* Quand un objet est à l'état OK, il est vérifié toutes les `checkinterval` minutes. \* S'il passe à l'état WARNING, CRITICAL ou UNKNOWN, il sera alors vérifié `maxcheckattempts` fois à un intervalle de `retry_interval` minutes avant de déclencher une notification. On considère alors que le service testé ou l'hôte est à l'état hard. D'autres périodes sont définies dans le fichier `$ETC/templates/time_templates.cfg`. Il y a notamment la période `1hourlong` qui est intéressant pour la suite car elle est utilisée par le service `Disks` ce qui va permettre de tester les notifications. `<code> # Check every 1hour with hard state after 6hours define service { name 1hourlong use generic-service maxcheckattempts 6 normalcheckinterval 60 retryinterval 60 register 0 } </code>` `<code shell> root@ctShinken:~# cat /etc/shinken/packs/linux-snmp/services/disks.cfg define service { servicedescription Disks use 1hourlong,linux-service register 0 hostname linux-snmp checkcommand checklinuxdisks DETAILEDDESC Overall disks usage _IMPACT Depends on disks, cause system instability _FIXACTIONS Clean the appropriate disks } </code>` Le service `Disks` redéfinit donc à son niveau quelques valeurs de `generic_service`. 3 paramètres sont particulièrement intéressants : \* `maxcheckattempts 6` → Nombre de "check" renvoyant une erreur avant l'envoi d'une notification \* `normalcheckinterval 60` → temps entre 2 "check" renvoyant une erreur \* `retryinterval 60` → temps entre 2 "check" ne renvoyant pas d'erreur Les temps sont exprimés en minutes. Si on applique cela, la première notification nous parviendra après un temps calculé avec la formule suivante : \* `normalcheckinterval + (retryinterval * (maxcheckattempts - 1))` moins 1 parce que le premier retour avec erreur compte. Donc en appliquant les valeurs du modèle `1hour_long` on obtient : \*  $60 + (60 * (6 - 1)) = 360$  minutes soit 6 heures Ce qui est beaucoup trop pour nos tests. On va donc utiliser plutôt le modèle de période suivant : `<code> define service { name 5minshort use generic-service maxcheckattempts 1 normalcheckinterval 5 retryinterval 2 register 0 } </code>` En modifiant ainsi le service `Disks` de notre modèle `linuxsnmp`. `<code> define service { servicedescription Disks use 5minshort,linux-service register 0 hostname linux-snmp checkcommand checklinuxdisks _DETAILEDDESC Overall disks usage _IMPACT Depends on disks, cause system instability _FIXACTIONS Clean the appropriate disks } </code>` A ce stade les alertes sont déclenchées dès la première erreur après un temps calculé avec la formule suivante : \*  $5 + (2 * (1 - 1)) = 5$  minutes. Puis ensuite toutes les 2 minutes. ===== Configuration du déclenchement des notifications ===== Une alerte déclenchée n'est pas pour autant notifiée. L'autorisation de notification pour un service ou un hôte, la plage de notification et la fréquence de notification est définie de façon générale dans les modèles `generichost` et `generic-service`. La directive `notification_interval` est modifiée ci-dessous pour envoyer des messages toutes les 10 minutes dans les tests (par défaut 1440 soit 24 heures). Les contacts notifiés et le niveau d'erreur à partir duquel une notification est envoyée est aussi stipulé ici. `<code> # Generic service definition template - This is NOT a real service, just a template! define service{ name generic-service ; The 'name' of this service template activechecksenabled 1 ; Active service checks are enabled passivechecksenabled 1 ; Passive service checks are enabled/accepted notificationenabled 1 ; Service notifications are enabled notificationinterval 10 notificationperiod 24x7 ... notificationoptions w,u,c,r ; Send notifications about warning, unknown, critical, and recovery events contactgroups admins,users stalkingoptions o,w,u,c register 0 ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL SERVICE, JUST A TEMPLATE } </code>` Attention, il faut bien sûr préciser dans le fichier `$ETC/hosts/localhost.cfg` que l'on veut notifier notre contact en cas de problème. `<code> define host { use linux-snmp,generic-host contactgroups admins hostname localhost address localhost } </code>` Dans notre cas, le groupe `admins` étant notifié, on va rajouter dans ce groupe l'utilisateur `roger` : `<code shell> root@ctShinken:~# cat /etc/shinken/contactgroups/admins.cfg define contactgroup{ contactgroup_name admins alias admins members admin,roger } </code>` Après toutes ces modifications dans les fichiers de configuration, il faut faire un contrôle de ces fichiers (`service shinken check`) et relancer `Shinken`. ===== Je reviens au menu Shinken =====

- [Supervision des services avec Shinken](#)

From:  
/ - Les cours du BTS SIO

Permanent link:  
[/doku.php/reseau/supervision/shinken\\_11](/doku.php/reseau/supervision/shinken_11)

Last update: 2016/11/23 00:13

