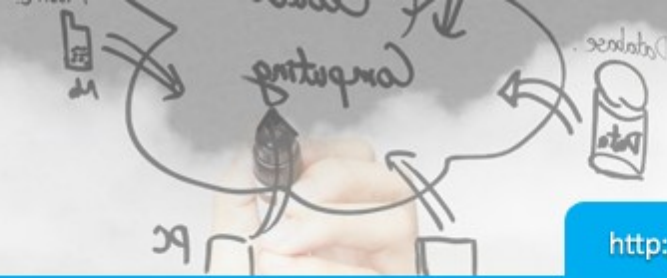




Supervision iRODS avec Nagios

Jérôme Pansanel et Emmanuel Medernach

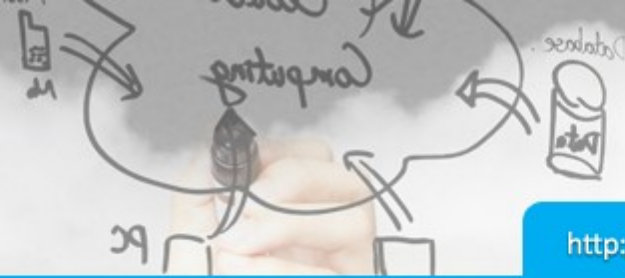
29 juin 2021



Objectifs

Les objectifs

- Compiler l'outil iping qui permet de superviser les serveurs iCAT et les serveurs de ressource
- Déployer les règles sur un serveur Nagios



iping

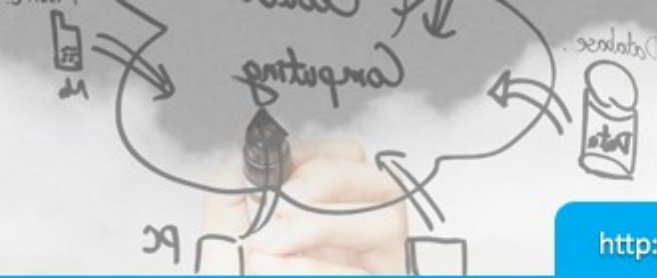
La commande iping

- Une commande à compiler pour déterminer si un serveur est en fonction ou pas
- **iping** exécute un appel `rcConnect()` pour déterminer l'état du serveur
- Le code source est disponible sur :
<https://github.com/irods/contrib/tree/master/iping>
- Mais nous utiliserons celui-là (branche iping-4.2.8) :
<https://github.com/Pansanel/contrib/tree/master/iping>

Construire le RPM

La commande iping

```
$ sudo yum install -y gcc gcc-c++ git openssl-devel rpm-build irods-devel \  
irods-externals-clang6.0-0 irods-externals-cmake3.11.4-0  
$ mkdir git && cd git  
$ git clone -b iping-4.2.8 https://github.com/Pansanel/contrib.git  
$ cd contrib  
$ git branch  
* iping-4.2.8  
$ mkdir ipingbuild && cd ipingbuild  
$ export PATH=/opt/irods-externals/cmake3.11.4-0/bin:$PATH  
$ cmake ~/git/contrib/iping  
$ make package  
...  
CPack: - package: /home/centos/git/contrib/ipingbuild/irods-iping-4.2.8-  
1.x86_64.rpm generated.  
$ sudo yum install -y /home/centos/git/contrib/ipingbuild/irods-iping-  
4.2.8-1.x86_64.rpm
```



Fonctionnement de iping

Une fois le paquet *iping* installé

- Tester la commande **iping** :

```
# Usage: iping [-h <host>] [-p <port>]
$ iping icat-X.novalocal
OK : connection to iRODS server successful
$ iping -h resourcel-X.novalocal
OK : connection to iRODS server successful
$ iping -h resource2-X.novalocal
OK : connection to iRODS server successful
$ echo $?
0
```

- Arrêter le service **irods** sur l'une des ressources (par ex. *resource1-X*)

```
$ iping -h resourcel-X.novalocal
ERROR: _rcConnect: connectToRhost error, server on resourcel-
X.novalocal:1247 is probably down status = -305111 USER_SOCKET_CONNECT_ERR,
Connection refused
$ echo $?
2
```


Un autre moyen

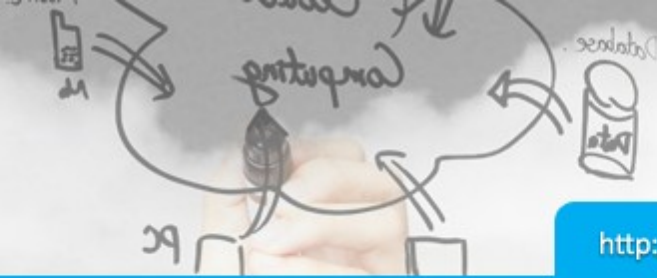
Une simple connexion peut suffire

```
$ yum install -y nmap-ncat
$ echo "CONNECT" | nc icat-X.novalocal 1247
<MsgHeader_PI>
<type>RODS_VERSION</type>
<msgLen>186</msgLen>
<errorLen>0</errorLen>
<bsLen>0</bsLen>
<intInfo>0</intInfo>
</MsgHeader_PI>
<Version_PI>
<status>-4000</status>
<relVersion>rods4.2.8</relVersion>
<apiVersion>d</apiVersion>
<reconnPort>0</reconnPort>
<reconnAddr></reconnAddr>
<cookie>400</cookie>
</Version_PI>
```

Configuration Nagios

Une configuration simple, mais utile

- Le service Nagios teste la ressource avec la commande **iping**
- Si un changement d'état est détecté, la valeur du paramètre de ressource **resc_status** est mis à jour pour refléter l'état de la ressource
- La valeur du paramètre **resc_status** est important pour le vote (par ex. dans le cas de la réplication)



Configuration Nagios

Points importants pour Nagios

- Un compte utilisateur iRODS doit être créé pour le service iRODS, avec les autorisations adéquates :
 - l'utilisateur *nagios* doit pouvoir avoir un shell
 - Il faut exécuter **iinit** en tant que *nagios*
- Mise à jour de la configuration Nagios pour définir les hôtes, les commandes et les services
- Créer les scripts qui seront utilisés pour surveiller les ressources et mettre à jour le paramètre **resc_status** :
 - `iping.sh` : le plugin nagios
 - `update_irods_resource_state.sh` : mise à jour du paramètre **resc_status**
- Tester le service !

Script `iping.sh`

```
#!/bin/bash

export HOME=/var/lib/nagios

return=0
/usr/bin/iping "$@" 2>&1 || return=$?

if [ $return -gt 3 ]; then
    exit 2
else
    exit $return
fi
```

Script update_irods_resc_state.sh

```
#!/bin/bash
export HOME=/var/lib/nagios
LOGFILE=/tmp/update_resc.log
echo update_irods_resc_state.sh "$@" >> $LOGFILE
HOST=$1
SERVICE_STATE=$2
SERVICE_STATE_TYPE=$3
SERVICE_ATTEMPT=$4
RESOURCES=$(iquest "%s" "select RESC_NAME where RESC_LOC = '$HOST'")
echo RESOURCES = $RESOURCES >> $LOGFILE
echo SERVICES_STATE = $SERVICE_STATE >> $LOGFILE
case "$SERVICE_STATE" in
OK)
    for RESOURCE in $RESOURCES; do
        iadmin modresc $RESOURCE status up
    done
    ;;
WARNING)
    ;;
UNKNOWN)
    ;;
CRITICAL)
    for RESOURCE in $RESOURCES; do
        iadmin modresc $RESOURCE status down
    done
    ;;
esac
exit 0
```

Assemblage des composants

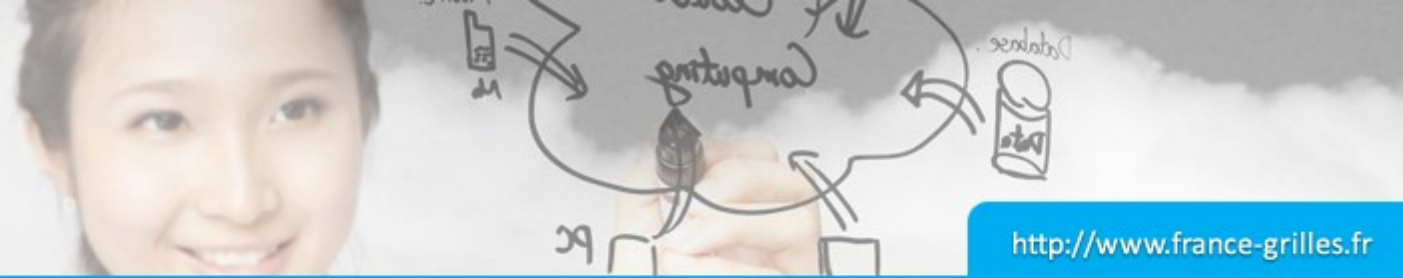
La configuration des services

```

define command {
    command_name      update-irods-resource-state
    command_line      /usr/lib64/nagios/plugins/update_irods_resc_state.sh
    $HOSTADDRESS$ $SERVICESTATE$ $SERVICESTATETYPE$ $SERVICEATTEMPT$
}

define service {
    use                generic-service
    hostgroups         irods-resource-servers
    service_description IPING
    check_command      iping-irods-server!1247
    check_interval     1
    event_handler      update-irods-resource-state
}

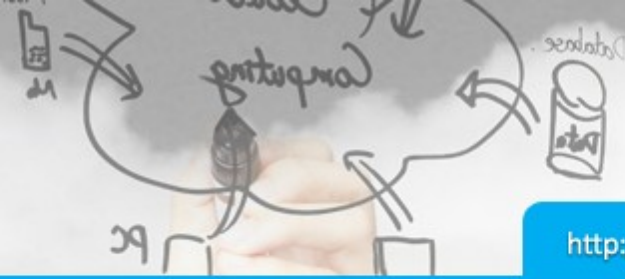
```



Pour aller plus loin

Quelques exemples

- Suivre le taux de remplissage des ressources
- Surveiller le nombre de connexions (commande **irods-grid**)
- Tester le cycle de vie d'une donnée
- Quelques liens :
 - <https://slides.com/jasoncposky/birmingham-monitoring-irods-with-nagios>
 - <https://github.com/FranceGrilles/monitoring-irods/tree/master/plugins>



Questions ?