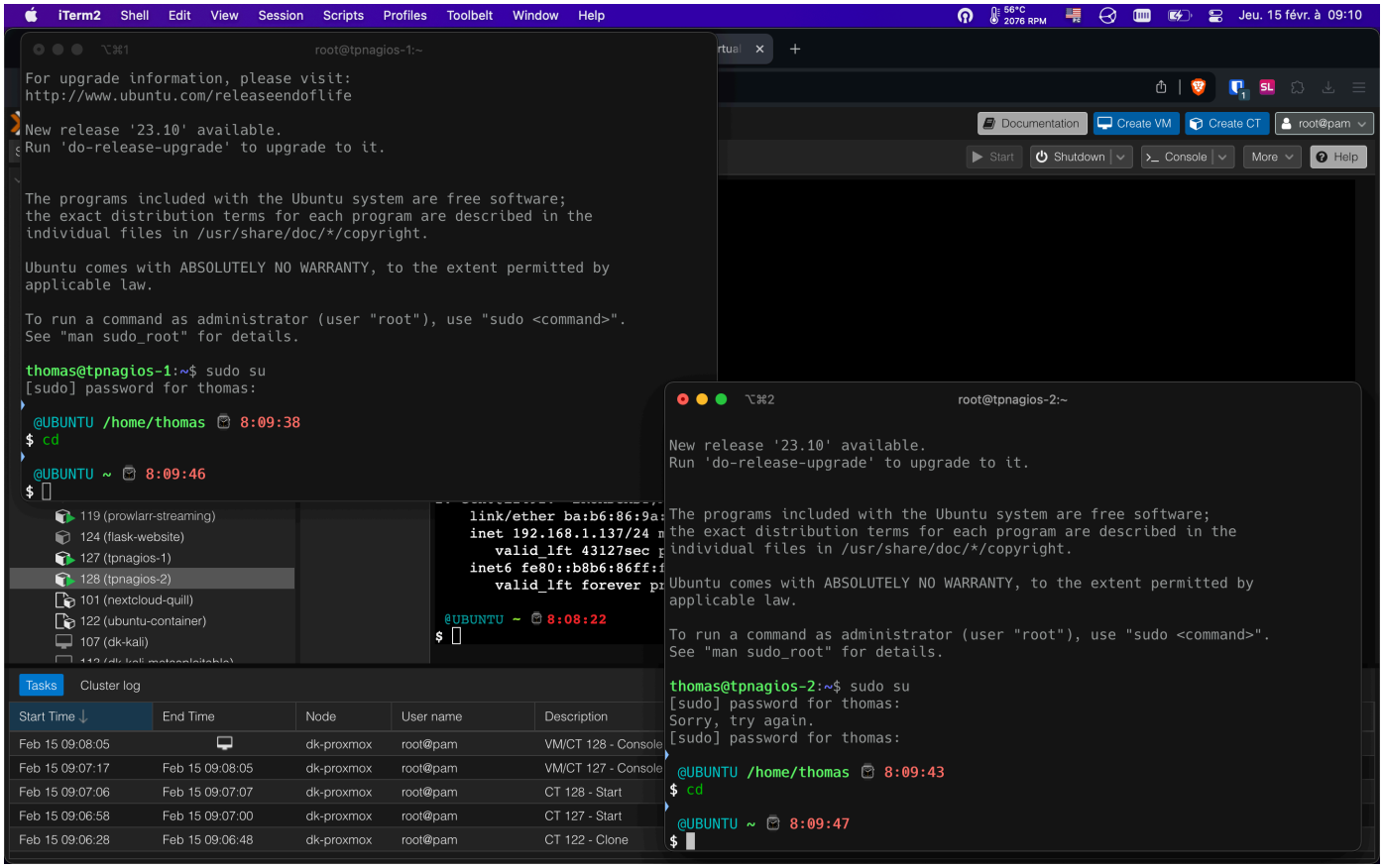
Supervision - Rendu TP01

TP effectué par `Thomas PEUGNET` et `David TEJEDA`.

Introduction

Préparation des VMs.



Script d'installation

Voici le script d'installation commenté.

```bash
#!/bin/bash

# Check if script is run as root, exit if not
if [ ! $UID -eq 0 ]
then
echo "Script must be run as root"
exit 1
fi

# Add nagios user and nagcmd group
useradd nagios
groupadd nagcmd

# Add nagios and www-data users to nagcmd group
usermod -a -G nagcmd nagios
usermod -a -G nagcmd www-data
```

```
# Create temporary directory and navigate to it
TMP_DIR=$(mktemp -d)
cd $TMP_DIR

# Download Nagios core software
wget http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.1.1.tar.gz

# Extract Nagios package
tar xvzf nagios-4.1.1.tar.gz
cd nagios-4.1.1

# Configure and compile Nagios core
./configure --with-command-group=nagcmd
make all

# Install Nagios, init script, config files, and command mode
make install
make install-init
make install-config
make install-commandmode

# Install Nagios web configuration for Apache
make install-webconf

# Create admin user for Nagios web interface
htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

# Reload Apache to apply changes
/etc/init.d/apache2 reload

cd $TMP_DIR

# Download Nagios plugins
wget http://www.nagiosplugins.org/download/nagios-plugins-2.1.1.tar.gz

# Extract plugins package
tar xvzf nagios-plugins-2.1.1.tar.gz
cd nagios-plugins-2.1.1

# Configure and compile Nagios
./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
make install
```

Installation de Nagios

Ayant rencontré de nombreux problèmes lors de l'installation du script, nous avons utilisé `docker` pour faire tourner notre instance.

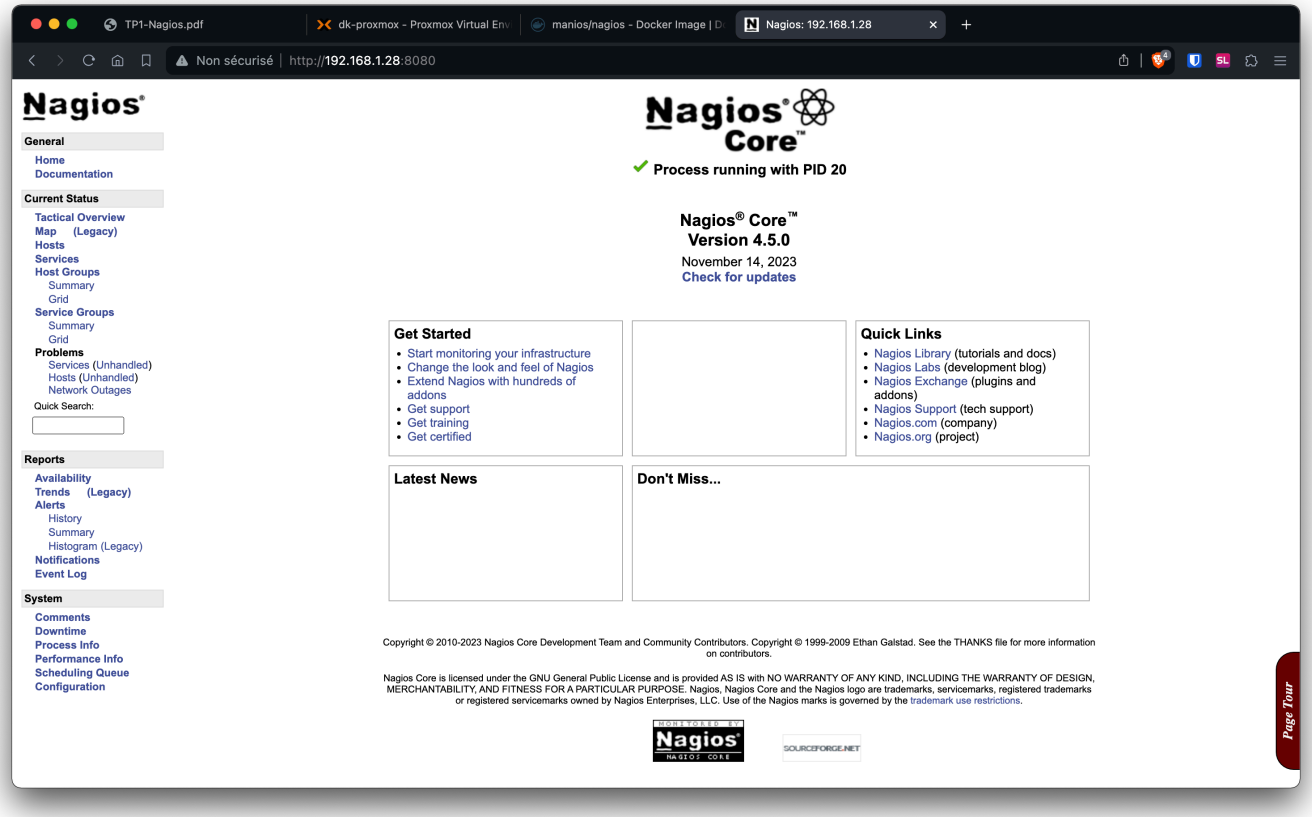
Nous avons donc créé les différents dossiers nécessaires au fonctionnement de Nagios.

```
$ tree nagios

nagios
|-- custom-plugins
|-- etc
|-- var

3 directories, 0 files
```

Puis nous avons lancé notre image `docker` avec un ensemble de paramètres.

```
$ docker run --name nagios  \
  -v /root/nagios/etc/:/opt/nagios/etc/ \
  -v /root/nagios/var:/opt/nagios/var/ \
  -v /root/nagios/custom-plugins:/opt/Custom-Nagios-Plugins \
  -p 0.0.0.0:8080:80 \
  manios/nagios:latest
```

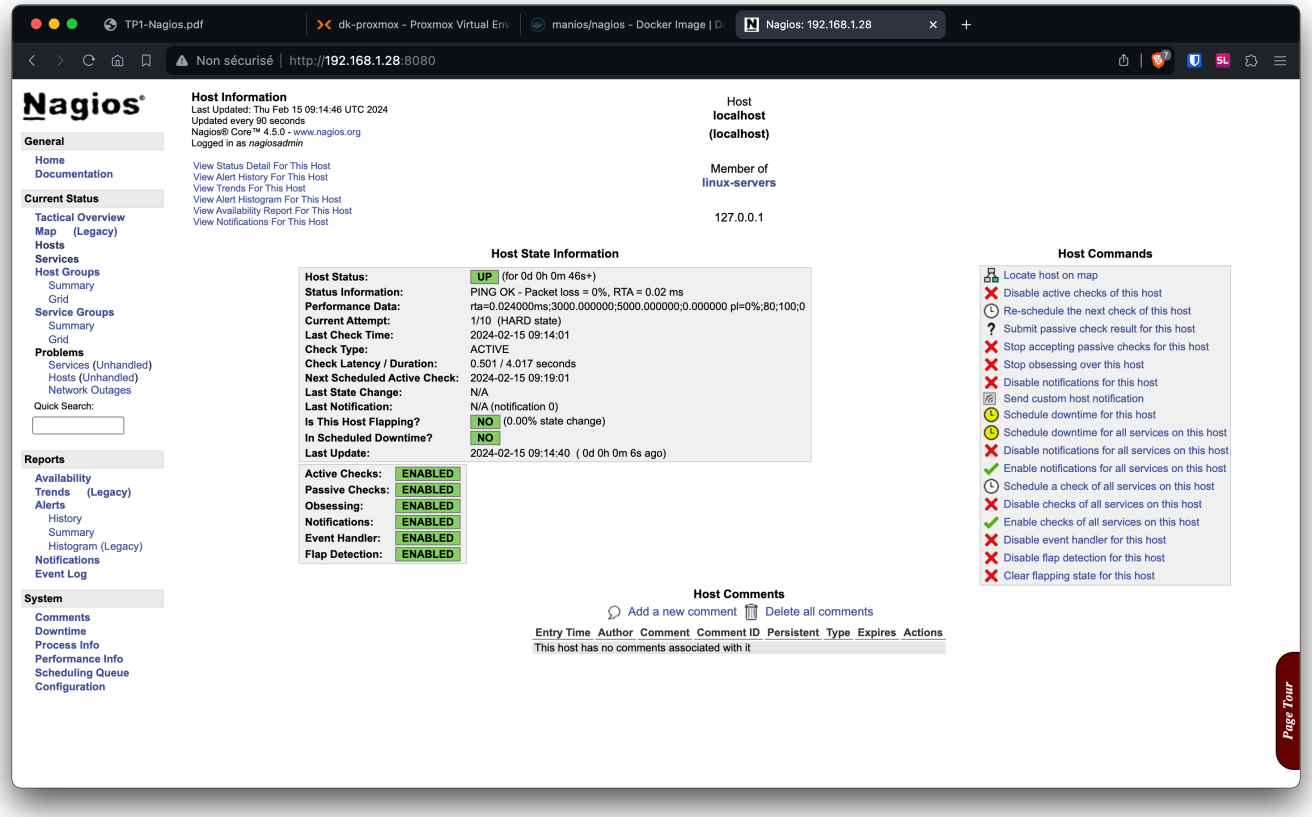
Une fois que l'instance est lancée, nous nous rendons sur notre navigateur à `http://192.168.1.28:8080`, étant l'adresse de notre conteneur `proxmox`.



Visiblement, ça fonctionne correctement.

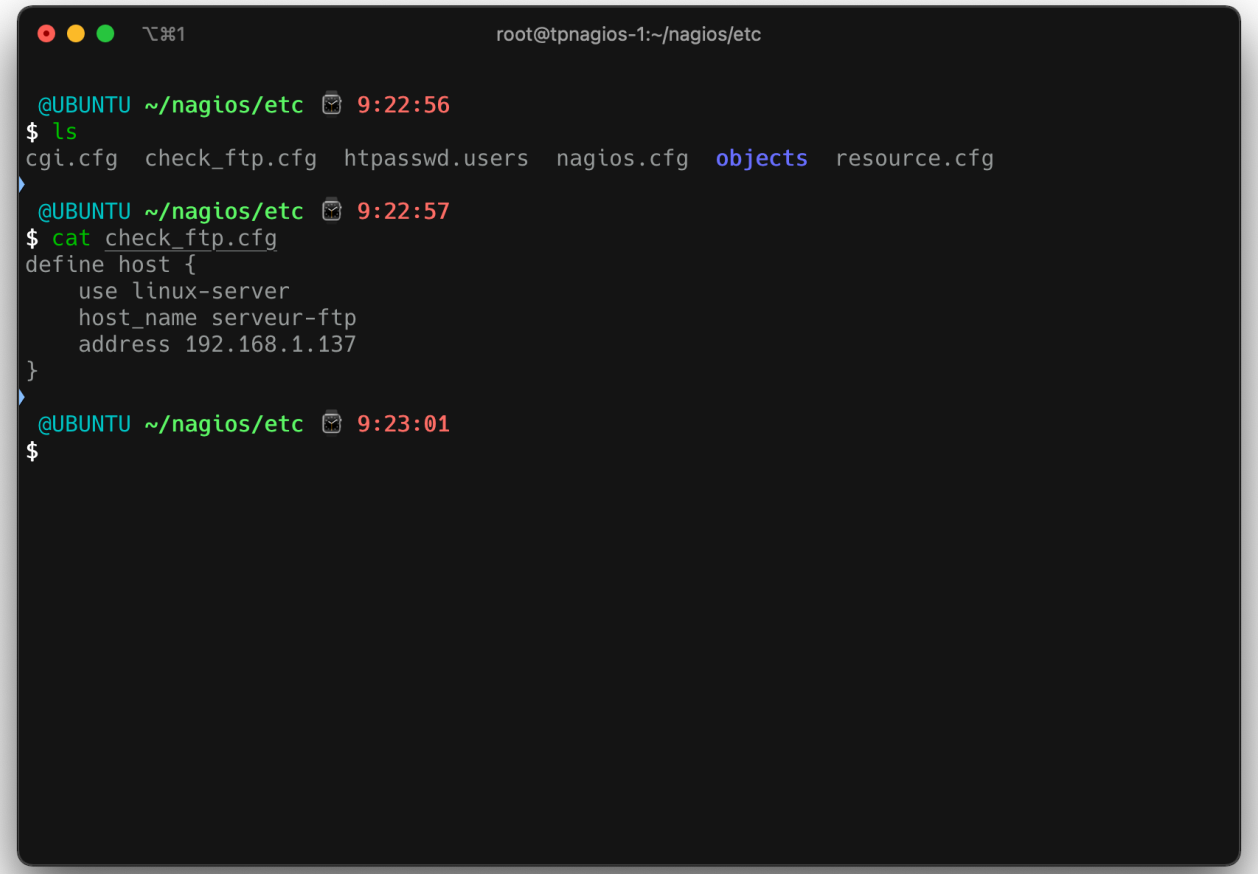
Machine supervisée par défaut

La machine supervisée par défaut est `localhost`, à savoir le serveur sur lequel tourne l'instance de Nagios.



Ajout de la supervision d'un serveur `ftp`

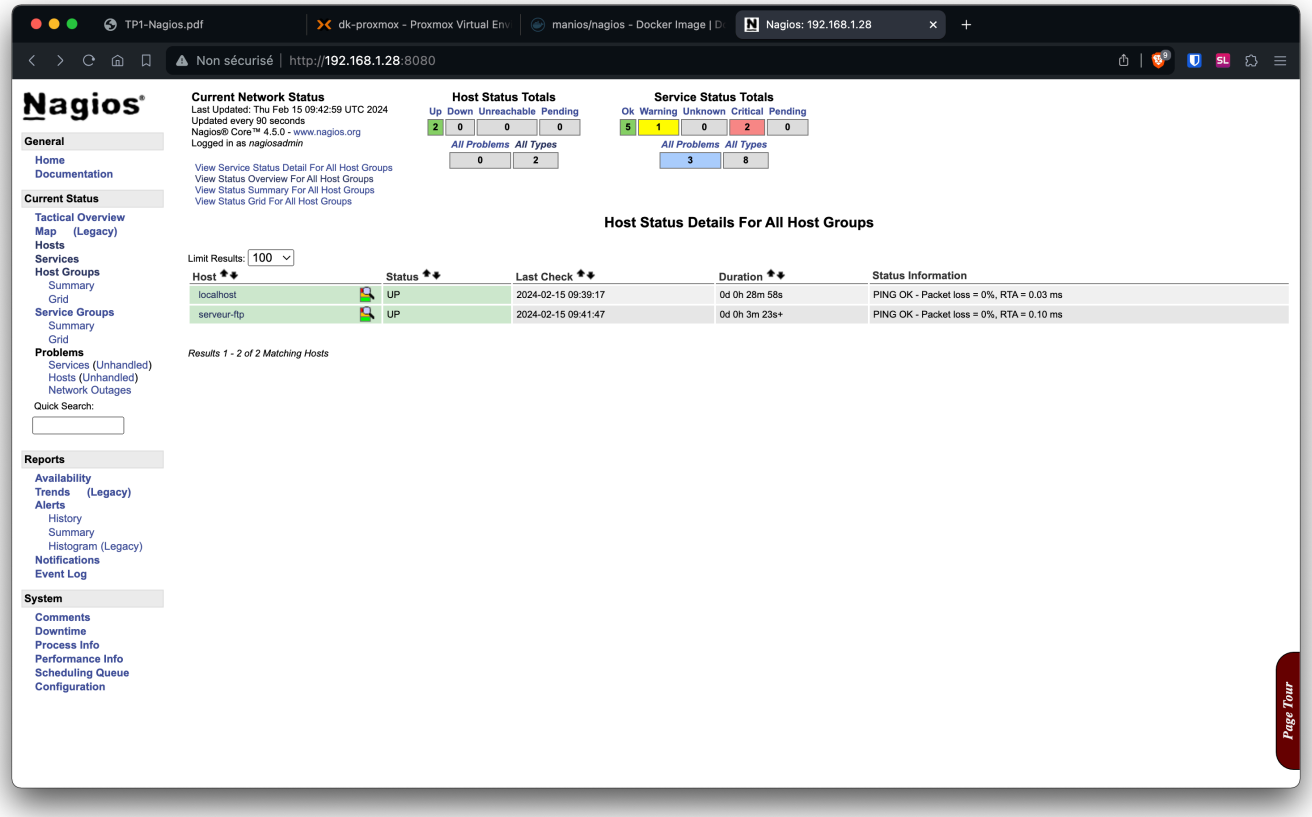
Nous ajoutons le fichier de configuration pour monitorer notre serveur `ftp`, avec `192.168.1.137` l'adresse de notre second conteneur Proxmox.



Après avoir déplacé le fichier dans le dossier `objects`, puis renommé en `ftp.cfg`, nous modifions le fichier de configuration `nagios.cfg` en ajoutant notre fichier de configuration.

```
⌥⌘3                                vim nagios.cfg
 19
 20
 21
 22 # OBJECT CONFIGURATION FILE(S)
 23 # These are the object configuration files in which you define hosts,
 24 # host groups, contacts, contact groups, services, etc.
 25 # You can split your object definitions across several config files
 26 # if you wish (as shown below), or keep them all in a single config file.
 27
 28 # You can specify individual object config files as shown below:
 29 cfg_file=/opt/nagios/etc/objects/commands.cfg
 30 cfg_file=/opt/nagios/etc/objects/contacts.cfg
 31 cfg_file=/opt/nagios/etc/objects/timeperiods.cfg
 32 cfg_file=/opt/nagios/etc/objects/templates.cfg
 33 cfg_file=/opt/nagios/etc/objects/ftp.cfg
 34
 35 # Definitions for monitoring the local (Linux) host
 36 cfg_file=/opt/nagios/etc/objects/localhost.cfg
 37
 38 # Definitions for monitoring a Windows machine
 39 #cfg_file=/opt/nagios/etc/objects/windows.cfg
 40
nagios.cfg                                          33,30          1%

-- VISUAL LINE --                                    1
```

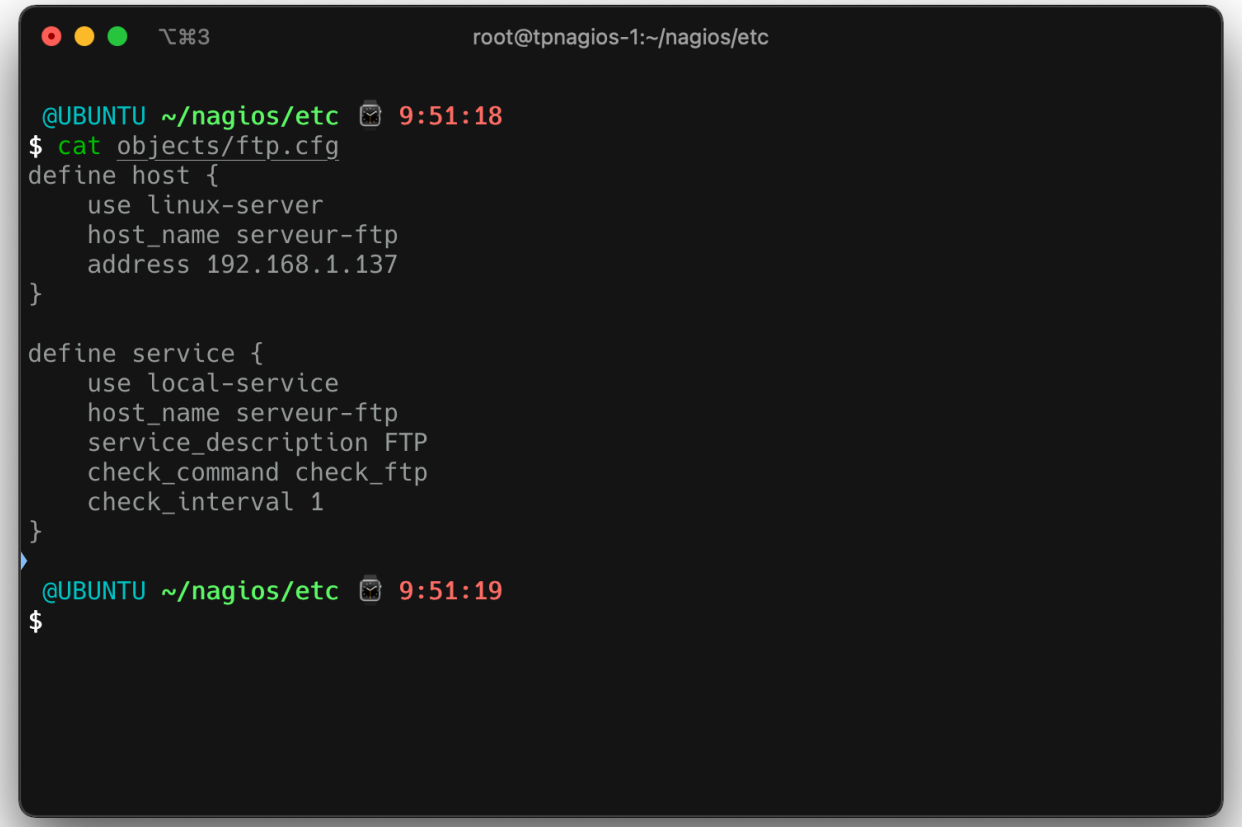
Après avoir relancé le service, nous obtenons le résultat suivant.



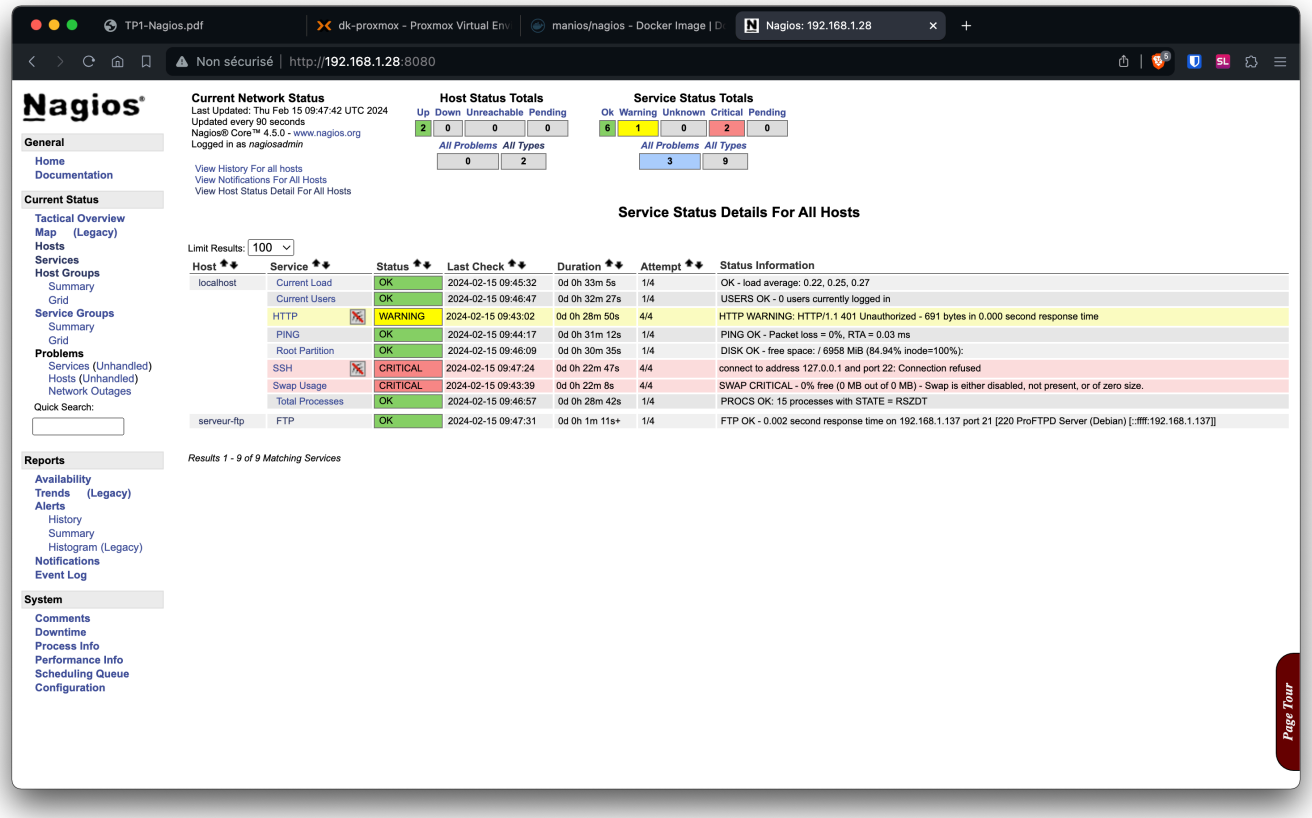
Nous pouvons constater que notre hôte `serveur-ftp` est bien détecté et en status `UP`.

Assignation du service `check_ftp`

Nous modifions maintenant notre fichier `ftp.cfg` pour ajouter l'assignation du service `check_ftp`.



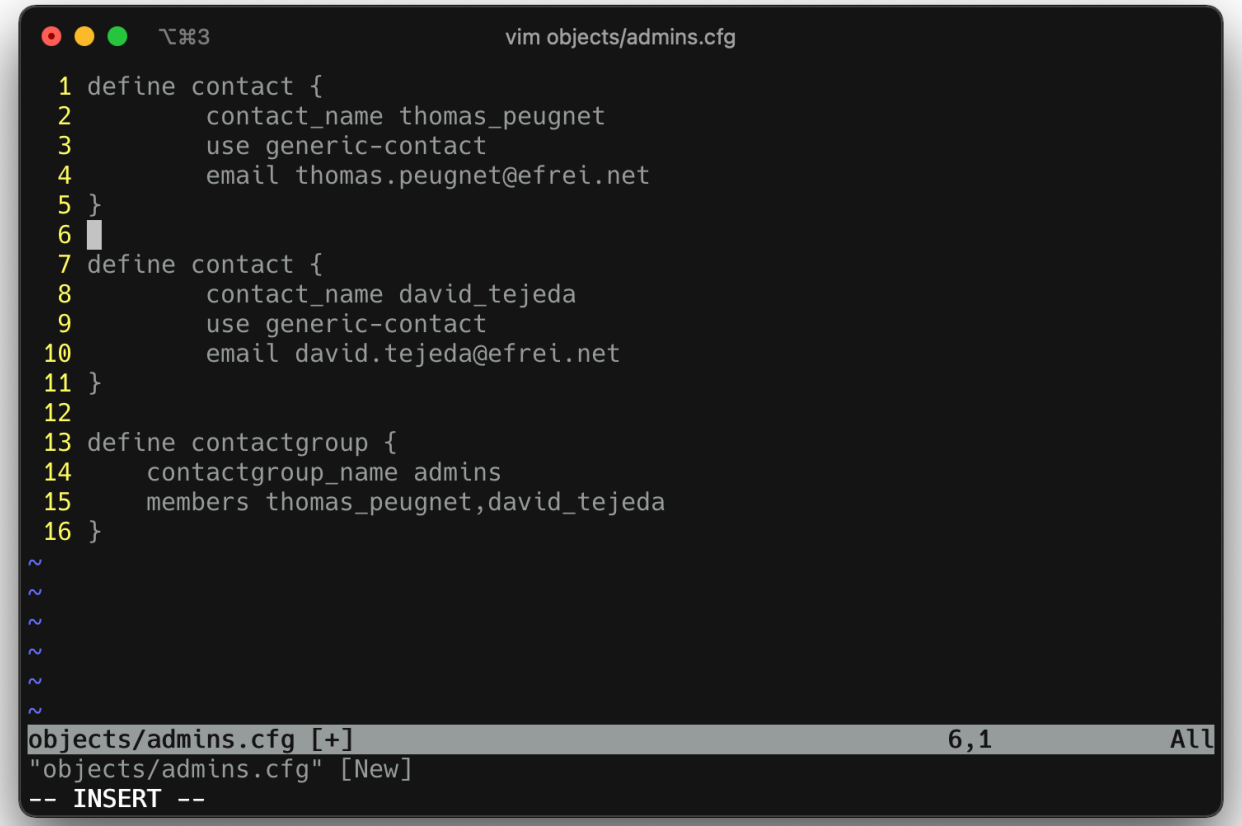
Après avoir relancé notre instance Nagios, nous constatons que le service est passé du status `pending` au status `up`.



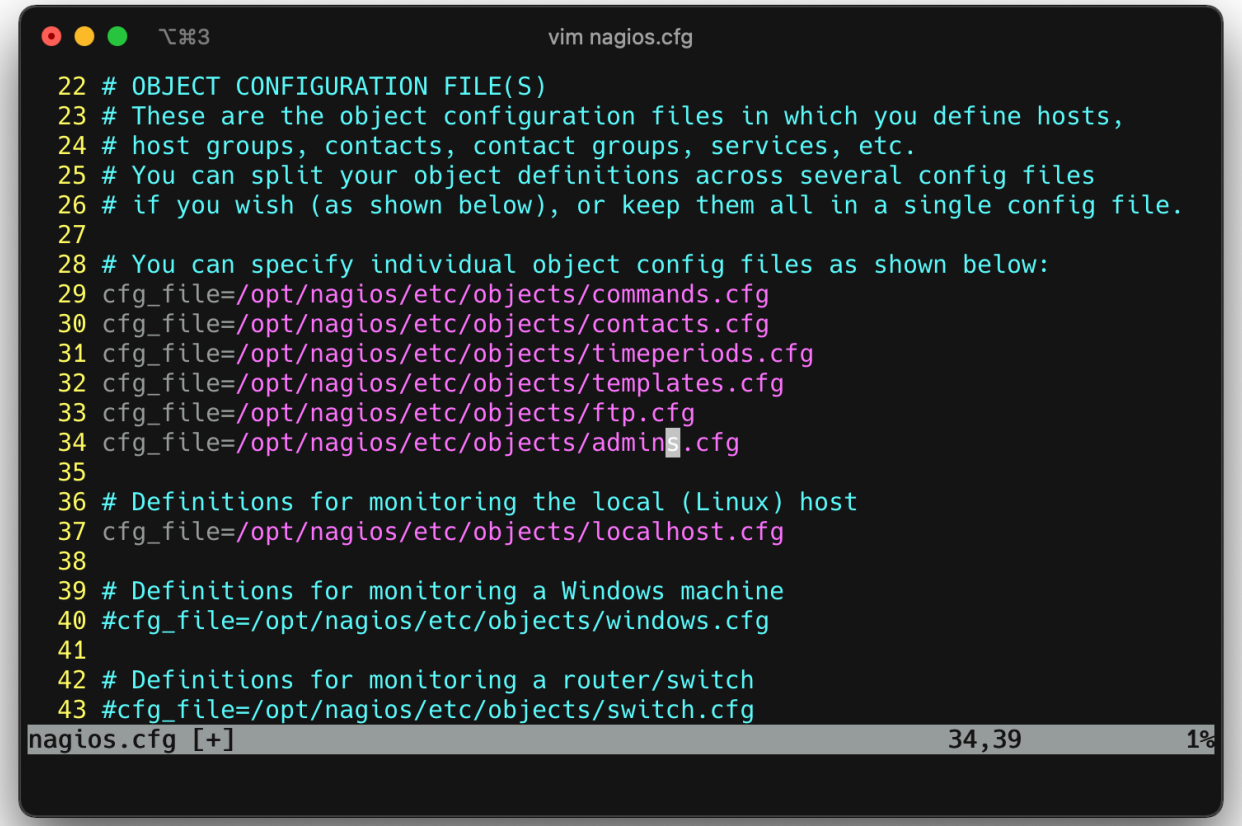
A noter que nous pouvons observer certains services en `CRITICAL`. Ceci est lié à l'utilisation d'une image `docker` n'ayant pas toutes les permissions ou les ports ouverts requis.

Notifications

Création des utilisateurs et des groupes pour obtenir des notifications.

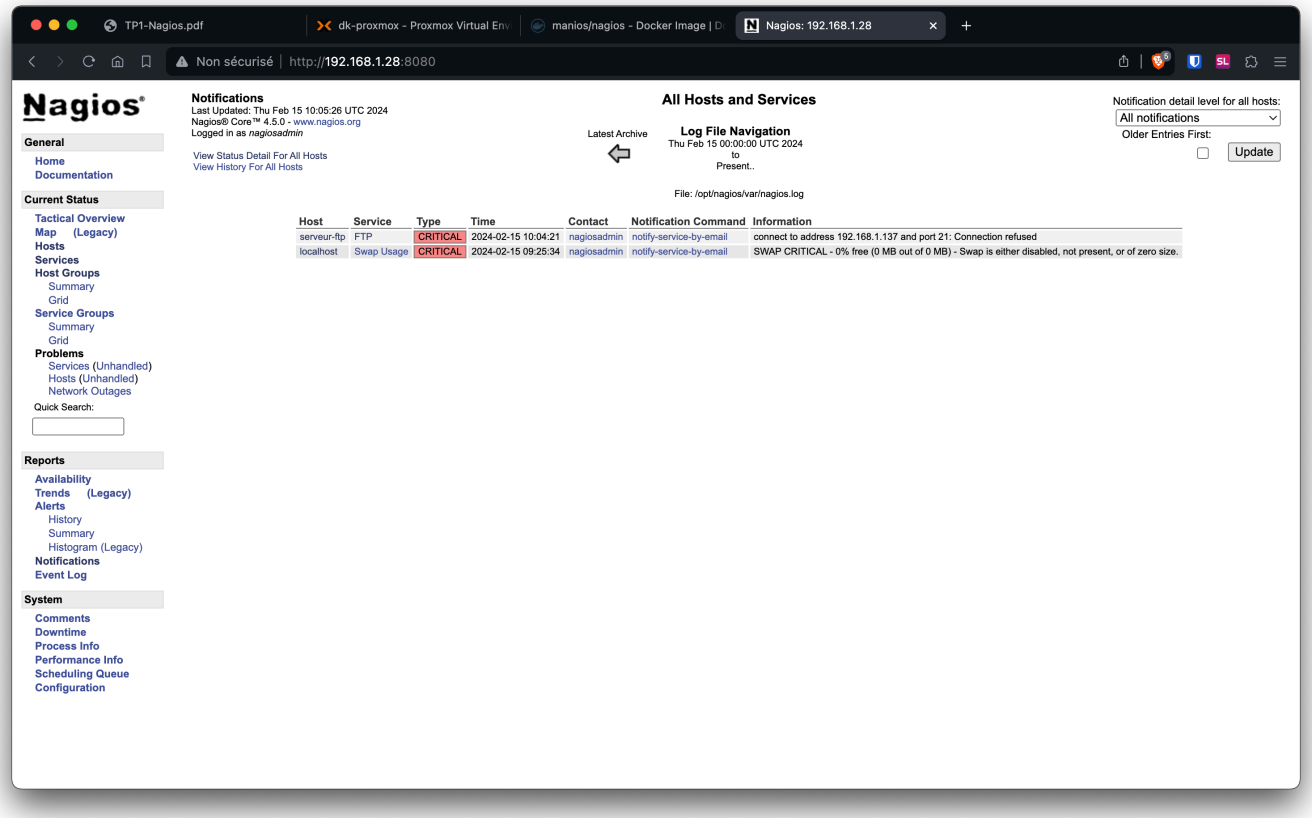


On modifie maintenant notre fichier de configuration `nagios.cfg` pour intégrer le nouveau fichier `admins.cfg`

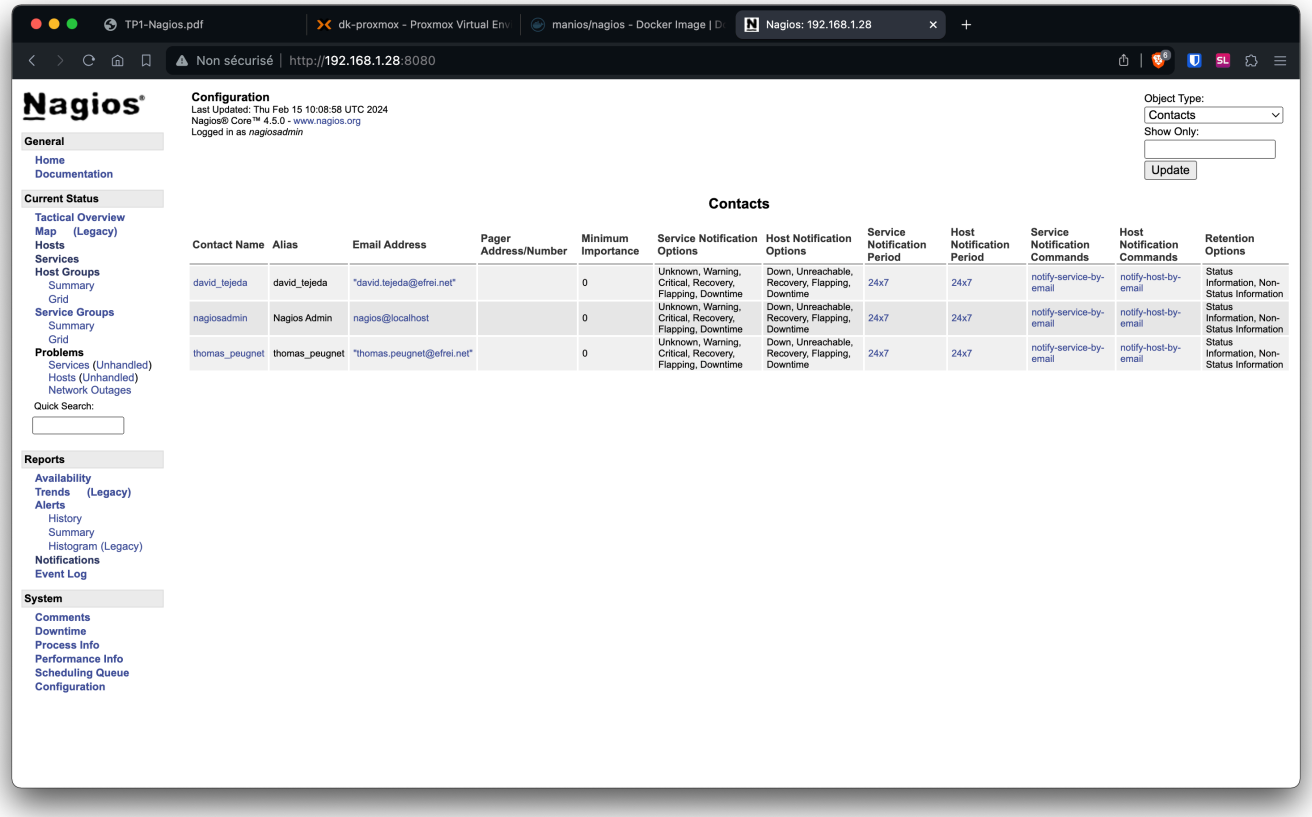


Arrêt du service `proftpd`

Après avoir arrêté le service `proftpd` sur le conteneur distant, nous constatons au bout de 90 secondes, que le service est passé en `CRITICAL`. Peu de temps après, nous voyons une nouvelle entrée dans la page Notifications.



En cliquant sur le le contact, nous trouvons la page suivante, indiquant que notre fichier de configuration avec les nouveaux contacts a bien été pris en compte.



Nous pouvons remarquer que, pour le moment, nous n'avons pas reçu de mail.

Supervision Avancée

Nous créons le script custom suivant.

```
#!/bin/bash

# Check args
if [ "$#" -ne 3 ]; then
    echo "Usage: $0 <IP address> <warning threshold in ms> <critical threshold in ms>"
    exit 3
fi

IP=$1
WARNING_THRESHOLD=$2
CRITICAL_THRESHOLD=$3

# 10 pings to the host
OUTPUT=$(ping -c 10 $IP)
if [ $? -ne 0 ]; then
    echo "Ping failed, unable to reach host"
    exit 3
fi

# Regexp for getting the output
JITTER=$(echo $OUTPUT | grep -oP '\d+.\d+/+\d+.\d+/+\d+.\d+/+\d+.\d+' | awk -F'/' '{print $4}')
JITTER_ROUNDED=$(echo $JITTER | awk '{print int($1)}')

# Comparer la gigue aux seuils et définir le code de retour
if [ $JITTER_ROUNDED -le $WARNING_THRESHOLD ]; then
    echo "OK - Jitter = ${JITTER_ROUNDED} ms"
    exit 0
elif [ $JITTER_ROUNDED -le $CRITICAL_THRESHOLD ]; then
    echo "Warning - Jitter = ${JITTER_ROUNDED} ms"
    exit 1
else
    echo "Critical - Jitter = ${JITTER_ROUNDED} ms"
    exit 2
fi
```

Nous créons donc un nouveau `check_jitter.sh` dans le dossier `/opt/Custom-Nagios-Plugins/`, et y ajoutons le contenu de notre script. Nous ajoutons les permissions d'exécution avec la commande `chmod +x check_jitter.sh`.

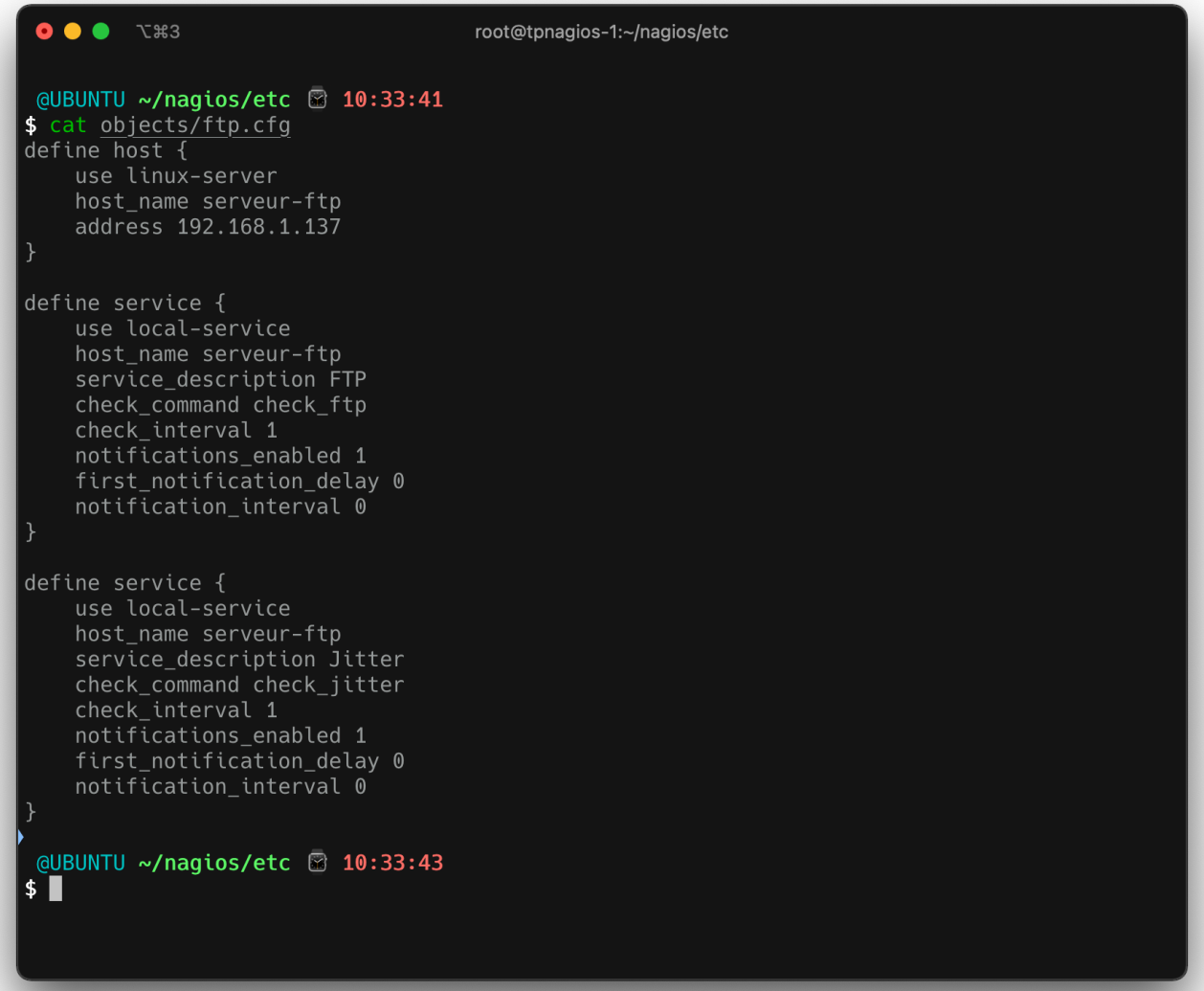
Ensuite, nous ajoutons à notre fichier `commands.cfg` la définition de la commande `check_jitter`.

```
define command {
  command_name check_jitter
  command_line /opt/Custom-Nagios-Plugins/check_jitter.sh $HOSTADDRESS$ $ARG1$ $ARG2$
}
```

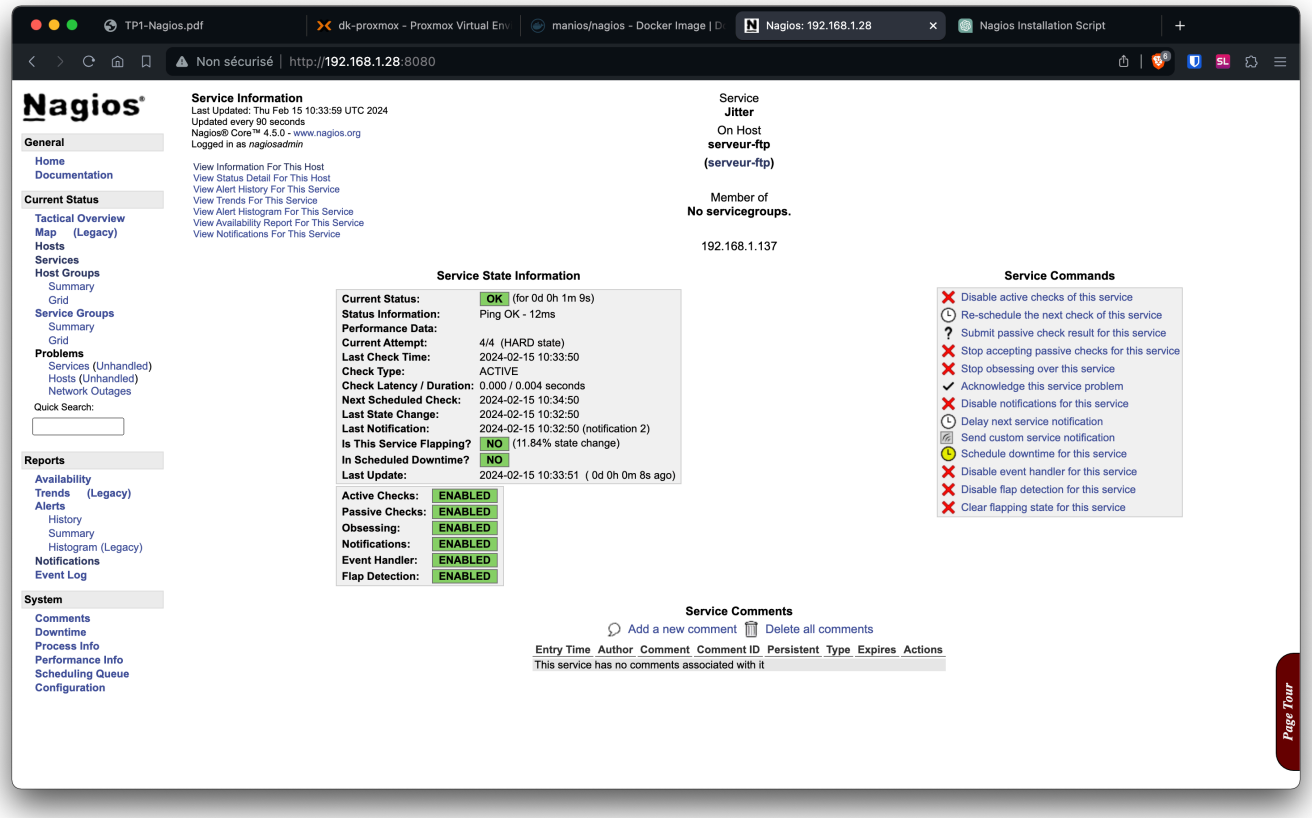
Enfin, nous créons une nouvelle assignation de service `check_jitter` à notre serveur FTP.

```
define service {
    use local-service
    host_name serveur-ftp
    service_description Jitter
    check_command check_jitter
    check_interval 1
    notifications_enabled 1
    first_notification_delay 0
    notification_interval 0
}
```

Notre fichier `ftp.cfg` est donc le suivant.



Nous allons vérifier que l'assignation au service fonctionne correctement depuis la WebUI.



Visiblement, le ping fonctionne correctement. En éteignant le conteneur du serveur FTP, nous obtenons le résultat suivant.