OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

Функциональные и нефункциональные требования

• ПК на Unix с 8ГБ ОЗУ или виртуальная машина с включенной Nested Virtualization.
• Созданный аккаунт на GitHub - https://github.com/
• Если Вы находитесь в России, для корректной работы Вам может потребоваться VPN.

Предварительно установленное и настроенное следующее ПО:
• Hashicorp Vagrant (https://www.vagrantup.com/downloads)
• Oracle VirtualBox (https://www.virtualbox.org/wiki/Linux_Downloads).
• Любой редактор кода, например Visual Studio Code, Atom и т.д.

Все дальнейшие действия были проверены при использовании Vagrant 2.2.19, VirtualBox v6.1.32. В лабораторной работе используются Vagrant boxes с CentOS 8 Stream (версия 20210210.0). В ОС отключён SElinux и firewalld.

Серьёзные отступления от этой конфигурации могут потребовать адаптации с вашей стороны.

Домашнее задание

Написать сервис, который будет раз в 30 секунд мониторить лог на предмет наличия ключевого слова. Файл и слово должны задаваться в /etc/sysconfig

Для начала создаём файл с конфигурацией для сервиса в директории /etc/sysconfig - из неё сервис будет брать необходимые переменные.

```
[root@nginx ~#] cat /etc/sysconfig/watchlog
# Configuration file for my watchlog service
# Place it to /etc/sysconfig

# File and word in that file that we will be monit
WORD="ALERT"
LOG=/var/log/watchlog.log
```

Затем создаем /var/log/watchlog.log и пишем туда строки на своё усмотрение, плюс ключевое слово ‘ALERT’

Создадим скрипт:

```
[root@nginx ~#] cat /opt/watchlog.sh
#!/bin/bash

WORD=$1
LOG=$2
DATE=`date`

if grep $WORD $LOG &> /dev/null
then
    logger "$DATE: I found word, Master!"
else
    exit 0
fi
```

Команда logger отправляет лог в системный журнал

Добавим права на запуск файла:

```
[root@nginx ~#] chmod +x  /opt/watchlog.sh
```

Создадим юнит для сервиса:

```
[Unit]
Description=My watchlog service

[Service]
Type=oneshot
EnvironmentFile=/etc/sysconfig/watchlog
ExecStart=/opt/watchlog.sh $WORD $LOG
```

Создадим юнит для таймера:

```
[Unit]
Description=Run watchlog script every 30 second

[Timer]
# Run every 30 second
OnUnitActiveSec=30
Unit=watchlog.service

[Install]
WantedBy=multi-user.target
```

Затем достаточно только стартануть timer:

[root@nginx ~#] systemctl start watchlog.timer

И убедиться в результате:

[root@nginx ~#] tail -f /var/log/messages

```
Feb 26 16:48:57 terraform-instance systemd: Starting My watchlog service...
Feb 26 16:48:57 terraform-instance root: Tue Feb 26 16:48:57 +05 2019: I found word, master!
Feb 26 16:48:57 terraform-instance systemd: Started My watchlog service.
Feb 26 16:49:27 terraform-instance systemd: Starting My watchlog service...
Feb 26 16:49:27 terraform-instance root: Tue Feb 26 16:49:27 +05 2019: I found word, master!
Feb 26 16:49:27 terraform-instance systemd: Started My watchlog service.
```

Из epel установить spawn-fcgi и переписать init-скрипт на unit-файл. Имя сервиса должно также называться.

Устанавливаем spawn-fcgi и необходимые для него пакеты:

[root@nginx ~#] yum install epel-release -y && yum install spawn-fcgi php php-cli mod_fcgid httpd -y

/etc/rc.d/init.d/spawn-fcgi - сам Init скрипт, который будем переписывать

Но перед этим необходимо раскомментировать строки с переменными в /etc/sysconfig/spawn-fcgi

Он должен получится следующего вида:

```
[root@nginx ~#]  cat /etc/sysconfig/spawn-fcgi
# You must set some working options before the "spawn-fcgi" service will work.
# If SOCKET points to a file, then this file is cleaned up by the init script.
#
# See spawn-fcgi(1) for all possible options.
#
# Example :
SOCKET=/var/run/php-fcgi.sock
OPTIONS="-u apache -g apache -s $SOCKET -S -M 0600 -C 32 -F 1 -- /usr/bin/php-cgi"
```

А сам юнит файл будет примерно следующего вида:

```
[root@nginx ~#]  cat /etc/systemd/system/spawn-fcgi.service
[Unit]
Description=Spawn-fcgi startup service by Otus
After=network.target

[Service]
Type=simple
PIDFile=/var/run/spawn-fcgi.pid
EnvironmentFile=/etc/sysconfig/spawn-fcgi
ExecStart=/usr/bin/spawn-fcgi -n $OPTIONS
KillMode=process

[Install]
WantedBy=multi-user.target
```

Убеждаемся что все успешно работает:

```
[root@nginx ~#] systemctl start spawn-fcgi

[root@nginx ~#] systemctl status spawn-fcgi
```

Дополнить юнит-файл apache httpd возможностью запустить несколько инстансов сервера с разными конфигами

Для запуска нескольких экземпляров сервиса будем использовать шаблон в конфигурации файла окружения (/usr/lib/systemd/system/httpd.service):

```
[Unit]
Description=The Apache HTTP Server
Wants=httpd-init.service
After=network.target remote-fs.target nss-lookup.target httpd-
init.service
Documentation=man:httpd.service(8)

[Service]
Type=notify
Environment=LANG=C
EnvironmentFile=/etc/sysconfig/httpd-%I                    #добавим параметр %I сюда
ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
ExecReload=/usr/sbin/httpd $OPTIONS -k graceful
# Send SIGWINCH for graceful stop
KillSignal=SIGWINCH
KillMode=mixed
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

В самом файле окружения (которых будет два) задается опция для запуска веб-сервера с необходимым конфигурационным файлом:

 # /etc/sysconfig/httpd-first
OPTIONS=-f conf/first.conf

 # /etc/sysconfig/httpd-second
OPTIONS=-f conf/second.conf

Соответственно в директории с конфигами httpd (/etc/httpd/conf) должны лежать два конфига, в нашем случае это будут first.conf и second.conf

Для удачного запуска, в конфигурационных файлах должны быть указаны уникальные для каждого экземпляра опции *Listen* и *PidFile*. Конфиги можно скопировать и поправить только второй, в нем должны быть след опции:

```
PidFile         /var/run/httpd-second.pid
Listen 8080
```

Этого достаточно для успешного запуска.

Запустим:

```
[root@nginx ~#] systemctl start httpd@first
[root@nginx ~#] systemctl start httpd@second
```

Проверить можно несколькими способами, например посмотреть какие порты слушаются:

```
[root@nginx ~#] ss -tnulp | grep httpd
```