В данной статье я расскажу о том, как настроить мониторинг почтового сервера postfix с помощью zabbix. В сети есть много советов и готовых инструкций, и моя в этом плане не будет какой-то уникальной. Я вначале хотел придумать что-то свое, но потом плюнул и воспользовался готовым скриптом, просто потому, что он полностью отвечает на вопрос. Но добавил к нему несколько дополнительных проверок и метрик. Я попробую более подробно раскрыть тему и дать несколько своих советов и рекомендаций.

Содержание:

Введение

Если у вас еще нет почтового сервера, то читайте как установить и настроить почтовый сервер postfix, а если нет сервера мониторинга, то соответственно вот это - установка и настройка zabbix на centos или то же самое на debian. Для мониторинга я буду использовать известную утилиту **pflogsumm**, которая анализирует почтовый лог postfix. В разных системах он может называться по-разному:

- /var/log/**mail.log** в debian и ubuntu

- /var/log/**maillog** в centos и freebsd

Само содержание будет одно и то же, поэтому статья будет актуальна для любого сервера с postfix и любой версией мониторинга zabbix. Я за основу возьму bash скрипт, который чаще всего попадается, если в поисковике поискать что-то на тему мониторинга postfix в заббиксе. Не знаю, кто у кого его скопировал изначально, поэтому автора указывать не буду. Сам скрипт достаточно простой, я расскажу подробно что он делает и как работает, чтобы вы понимали и по возможности могли его изменить так, как вам потребуется.

Я буду показывать на примере системы CentOS. Нам понадобится установить утилиту pflogsumm. Делается это просто.

```
# yum install postfix-pflogsumm
```

Подготовка сервера к мониторингу

Выполним некоторые подготовительные действия на сервере. Для начала настроим ротацию лога postfix.

Настройка ротации лога maillog

По-умолчанию, за ротацию почтового лога отвечает конфигурационный файл */etc/logrotate.d/syslog*. В нем не указан интервал, как часто происходит ротация, поэтому используется значение по-умолчанию из общего конфигурационного файла logrotate - */etc/logrotate.conf*. Там интервал указан равный неделе.

Если вы посмотрите на время ротации, то сходу не будет понятно, почему используется именно это время. В разных системах оно будет разным. Различия будут даже в разных версиях centos. Например, в 5-й версии ротация будет запускаться следующим скриптом для cron - */etc/cron.daily/logrotate*. Запускаться этот скрипт будет в соответствии с расписанием, указанным в файле */etc/crontab*.

```
# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
```

```
42 4 1 * * root run-parts /etc/cron.monthly
```

Таким образом, в CentOS 5 ротация лога maillog будет происходить раз в неделю в 4:02. Не самый удобный расклад. Теперь посмотрим, в какое время эта же операция производится в CentOS 7 и как ее изменить. Здесь в */etc/crontab* уже пусто, но есть другой файл */etc/cron.d/0hourly*, в котором указано расписание запуска скриптов hourly

```
01 * * * * root run-parts /etc/cron.hourly
```

То есть каждую минуту. Теперь посмотрим, что в директории */etc/cron.hourly*. Там есть файл *0anacron*, который отвечает за запуск anacron, который, в свою очередь, отвечает за запуск ежедневных, еженедельных и ежемесячных задач. Интересующая нас задача лежит в */etc/cron.daily/logrotate*. Посмотреть, когда она выполняется, можно в конфиге anacron - */etc/anacrontab.*

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days delay in minutes job-identifier command
1<----->5<----->cron.daily<---><------>nice run-parts /etc/cron.daily
7<----->25<---->cron.weekly<--><------>nice run-parts /etc/cron.weekly
@monthly 45<--->cron.monthly<--><------>nice run-parts /etc/cron.monthly
```

С такими данными получается, что ротация логов запускается, начиная с 3-х часов ночи, с максимальным сдвигом до 3:45 каждый день. То есть в любое время в этом промежутке. Ну а так как в самом logrotate указано, что ротация maillog происходит раз в неделю, она, соответственно, раз в неделю и происходит примерно в это время.

> Для тех, кто будет настраивать мониторинг postfix в debian, сразу скажу, что там ротация логов настроена точно так же, как в CentOS 7 через anacron. Но запускается он по-другому. Время его запуска явно указано в файле */etc/cron.d/anacron*, без лишних перенаправлений.

Я на распутывание этих клубков с ротацией логов тратил больше времени, чем на саму настройку мониторинга. Весь смысл в том, что мне кажется такая схема ротации неудобной. Особенно в нагруженных серверах. За неделю там такой лог огромный получается, что его невозможно быстро обработать. Мне больше нравится, когда логи ротируются раз в сутки примерно в 00:00. Так и статистику снимать удобнее, и анализировать логи. Настроим это.

Первым делом надо подправить конфиг **logrotate** и указать для maillog отдельные настройки с ежедневной ротацией. Делаем это.

```
# mcedit /etc/logrotate.d/syslog
```

```
/var/log/cron
/var/log/messages
/var/log/secure
/var/log/spooler
{
    missingok
    sharedscripts
    postrotate
        /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true
    endscript
}

/var/log/maillog
{
    daily
    missingok
    sharedscripts
```

```
    postrotate
        /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true
    endscript
}
```

Тут ничего придумывать не надо. Просто копируем в отдельный раздел настройки для */var/log/maillog* и добавляем туда параметр **daily**. В разных системах стандартный конфиг для syslog будет разный. Нам его не обязательно менять, одного дополнительного параметра достаточно.

Теперь нужно настроить запуск anacron в полночь. В CentOS 7 редактируем конфиг */etc/anacrontab*. Меняем 2 параметра:

```
RANDOM_DELAY=1
START_HOURS_RANGE=0-1
```

Параметры для cron.daily не трогаем. В итоге anacron будет запускать скрипты cron.daily в интервале с 0 до 1 часа, с задержкой 5 минут от полуночи и с максимальным разбросом этой задержки в 1 минуту. То есть примерно в 0:04-0:06.

В Debian 7 и 8 для такого же результата достаточно просто изменить время запуска anacron в файле */etc/cron.d/anacron*.

```
0 0    * * *   root test -x /etc/init.d/anacron && /usr/sbin/invoke-rc.d anacron start >/dev/null
```

Все. Настроили нормальную ротацию логов. Теперь можно обрабатывать эти логи и снимать статистику. Описанные выше настройки не обязательны, все будет работать и без них. Но так удобнее.

Проверка работы pflogsumm

Снимать параметры системы мы будем с помощью pflogsumm, поэтому для начала проверим, как она работает и какую информацию выводит. Для этого запускаем ее с такими ключами:

```
# pflogsumm -d yesterday -h 0 -u 0 --bounce_detail=0 --deferral_detail=0 --reject_detail=0 --smtpd_warning_detail=0 --
no_no_msg_size /var/log/maillog
```

Если получите ошибки на некоторые ключи, попробуйте вариант для более старой версии:

```
pflogsumm -d yesterday -h 0 -u 0 --no_bounce_detail --no_deferral_detail --no_reject_detail --no_smtpd_warnings --
no_no_msg_size /var/log/maillog
```

Вы должны получить примерно такой вывод:

```
Grand Totals
------------
messages

   1638   received
   2769   delivered
      0   forwarded
      2   deferred  (15  deferrals)
     36   bounced
    259   rejected (8%)
      0   reject warnings
      0   held
      0   discarded (0%)

    712m  bytes received
   1062m  bytes delivered
    840   senders
    445   sending hosts/domains
    243   recipients
    100   recipient hosts/domains


Per-Hour Traffic Summary
    time          received  delivered   deferred    bounced     rejected
    ------------------------------------------------------------------------
    0000-0100           47         85          0          5            8
    0100-0200           37         53          0          0            5
    0200-0300           41         54          0          7           57
    0300-0400           33         69          0          2            3
    0400-0500           42         84          0          0           26
    0500-0600           36         62          0          0           20
    0600-0700           49         63          0          0            6
    0700-0800           56         96          0          0            3
    0800-0900           63         83          0          1            3
    0900-1000           82        124          0          2            3
    1000-1100          154        276          0          6            4
    1100-1200          120        172          3          3           22
    1200-1300          161        264          2          6           10
    1300-1400          112        154          1          0            5
    1400-1500          101        183          1          0           13
    1500-1600           98        164          1          0           11
    1600-1700          109        153          1          1            7
    1700-1800          118        262          1          0            2
    1800-1900           59         94          1          0            4
    1900-2000           37        118          0          0            3
    2000-2100            9         13          1          0            4
    2100-2200           17         45          1          2           39
    2200-2300           14         26          1          0            0
    2300-2400           43         72          1          1            1

smtp delivery failures: none

Warnings
--------
  submission/smtpd (total: 79)
       27   186.7.129.52: hostname 52.129.7.186.f.dyn.claro.net.do verifica...
       18   171.4.249.187: hostname mx-ll-171.4.249-187.dynamic.3bb.co.th v...
        4   unknown[46.49.0.109]: SASL PLAIN authentication failed:
```

Я использую параметр **yesterday**, так как проверяю в системе, где еще не настроена ротация логов, и в maillog находится информация за последние несколько дней. Если у вас ночью была ротация файлов, то информации за вчера там уже нет, получите на выходе нули. Так что правильно указывайте либо параметр, либо лог файл.

Значения из первой части отчета мы и будем передавать в zabbix для сбора статистики. Если у вас нормально отрабатывает pflogsumm, можно дальше переходить к настройке мониторинга в zabbix.

Сразу сделаю важное замечание. Значения полученных и доставленных писем не будут соответствовать реальному количеству писем, если у вас есть какие-либо дополнительные обработчики писем, работа которых приводит к повторному появлению упоминания о письме в почте. К этому могут приводить антиспам системы, антивирусы, может быть что-то еще. У вас будут задваиваться или затраиваться упоминания в логе об одном и том же письме.

Эти значения имеют смысл, только если вам нужны абсолютные значения для статистики. Для мониторинга это не принципиально, так как важна будет динамика изменений, а не точные цифры. Мониторинг должен будет отреагировать на существенное увеличение количества писем относительно средних параметров за прошлые периоды. Каковы же были эти значения, не принципиально.

Настройка zabbix агента

Теперь напишем скрипт, который будет парсить вывод pflogsumm и передавать его в zabbix. Создадим папку для скриптов, если у вас ее нет и положим туда скрипт.

```
# mkdir /etc/zabbix/scripts
# mcedit /etc/zabbix/scripts/postfix.sh
```

```
#!/bin/bash

export LC_ALL=""
export LANG="en_US.UTF-8"
#
if [[ -z "$1" ]]; then
```

```
  exit 1
fi
##### PARAMETERS #####
MAILQ=$(which mailq)
PFLOGSUMM=$(which pflogsumm)
MAILLOG="/var/log/maillog"
METRIC="$1"
CACHE_TTL="1740"
CACHE_FILE="/tmp/zabbix.postfix.cache"
EXEC_TIMEOUT="4"
NOW_TIME=`date '+%s'`
##### RUN #####
if [ "$METRIC" = "queue" ]; then
  TEMP_DATA=`${MAILQ} 2>&1 | tail -n1`
  if echo "${TEMP_DATA}" | grep -iq "Mail queue is empty"; then
    echo 0
  elif echo "${TEMP_DATA}" | grep -iPq "in\s+\d+\s+request"; then
    echo "${TEMP_DATA}" | sed -e 's/.*in\s\+\([0-9]\+\)\s\+request.*/\1/gI' 2> /dev/null | head -n1
  else
    # Error
    echo 65535
  fi
  exit 0
else
  if [ -s "${CACHE_FILE}" ]; then
    CACHE_TIME=`stat -c"%Y" "${CACHE_FILE}"`
    DELTA_TIME=$((${NOW_TIME} - ${CACHE_TIME}))
    if [ ${DELTA_TIME} -lt ${EXEC_TIMEOUT} ]; then
      sleep $((${EXEC_TIMEOUT} - ${DELTA_TIME}))
    elif [ ${DELTA_TIME} -gt ${CACHE_TTL} ]; then
      echo "" >> "${CACHE_FILE}" # !!!
      DATE_TO=`date +%d\ %H:%M:%S`
      DATE_FROM=`date -d @${CACHE_TIME} +%d\ %H:%M:%S`
```

```
      DATA_CACHE=`sudo cat ${MAILLOG} | sed -e 's/^\([a-zA-Z]\{3\}\s\)\s\([0-9]\s\)/\10\2/g' | awk '$2" "$3>=from && $2"
"$3<=to' from="${DATE_FROM}" to="${DATE_TO}" | ${PFLOGSUMM} -h 0 -u 0 --bounce_detail=0 --deferral_detail=0 --
reject_detail=0 --smtpd_warning_detail=0 --no_no_msg_size 2>&1`
      echo "${DATA_CACHE}" > ${CACHE_FILE} # !!!
      chmod 640 ${CACHE_FILE}
    fi
  else
    echo "" > ${CACHE_FILE} # !!!
    exit 0
  fi
  awk "BEGIN{IGNORECASE=1} /${METRIC}/ {print \$1}" ${CACHE_FILE} | awk '{print $1}' | awk '/k|m/{p = /k/?1:2}{printf
"%d\n", int($1) * 1024 ^ p}' | head -n1
fi
exit 0
```

```
# chown root:zabbix /etc/zabbix/scripts/postfix.sh
# chmod 750 /etc/zabbix/scripts/postfix.sh
```

Данный скрипт берет текущее время, проверяет время последнего запуска скрипта, выделяет из почтового лога только тот интервал, что попадает в этот отрезок времени и анализирует его. Очень удобная задумка и простая реализация. Я изначально до такого не додумался и анализировал весь лог. Когда увидел этот скрипт, решил взять за основу для своего мониторинга.

Позволяем пользователю zabbix, от которого работает мониторинг, запускать команды из скрипта. Для этого запускаем **visudo** и добавляем следующие строки в */etc/sudoers*.

```
zabbix ALL=(ALL) NOPASSWD: /bin/cat /var/log/maillog
zabbix ALL=(ALL) NOPASSWD: /usr/sbin/pflogsumm
zabbix ALL=(ALL) NOPASSWD: /usr/bin/mailq
```

Далее создадим кэш файл для проверки работы скрипта с заведомо старой датой

```
# sudo -u zabbix touch /tmp/zabbix.postfix.cache
# echo "" >> /tmp/zabbix.postfix.cache
# sudo -u zabbix touch -m -d '2010-01-01 22:22' /tmp/zabbix.postfix.cache
```

Остановлюсь подробнее на этом кэше. В скрипте есть параметр **CACHE_TTL**, установленный в 1740 секунд = 29 минут. Это чуть меньше, чем интервал обновления элементов данных на сервере zabbix - 30 минут. Кэш нужен для того, чтобы лишний раз не выполнять работу скрипта в тех ситуациях, когда по какой-то причине, например во время отладки, вы постоянно проверяете значения. Если еще не прошло 29 минут с момента последнего запуска скрипта, данные будут взяты из кэш файла *zabbix.postfix.cache*. Имейте это ввиду. Если вы заходите изменить интервал обновления с 30 минут, на 15, то не забудьте изменить время кэша на 14,5 минут.

Сразу расскажу о куче подводных камней и нюансах при использовании данного скрипта на разных системах. Вам в любом случае необходимо будет все проверить, прежде чем запускать сбор статистики в zabbix. Первым делом проверьте вывод команды mailq. Например, в CentOS 7 и Debian 7 в случае пустой очереди он будет таким:

```
# mailq
Mail queue is empty
```

Под такой вывод настроен текущий пример скрипта. А, к примеру, в CentOS 5 у меня вывод такой:

```
# mailq
/var/spool/mqueue is empty
Total requests: 0
```

Скрипт в текущем виде уже не подходит. Ту часть, что отвечает за проверку очереди сообщений, нужно изменить и привести к такому виду:

```
TEMP_DATA=`${MAILQ} 2>&1`
```

```
if echo "${TEMP_DATA}" | grep -q "/var/spool/mqueue is empty"; then
```

Остальное не трогаем. Дальше может такая проблема возникнуть. В той же CentOS 5 mailq никак не хочет запускаться под пользователем zabbix. Получаю ошибку, даже если настраиваю права в sudoers.

```
# sudo -u zabbix mailq
can not chdir(/var/spool/mqueue/): Permission denied
Program mode requires special privileges, e.g., root or TrustedUser.
```

Не стал долго разбираться, как это исправить, так как сходу решения не придумал. Пришлось запускать zabbix-agent под пользователем root. Для этого в */etc/zabbix/zabbix_agentd.conf* укажите параметры:

```
AllowRoot=1
User=root
```

И перезапустите агент.

Вместо mailq можно использовать команду:

```
# postqueue -p
```

Я не разбирался подробно, в чем разница, но на некоторых системах mailq мне показывала пустую очередь, а postqueue корректно отображала ее. В общем, в каждом конкретном случае нужно разобраться отдельно и убедиться, что все корректно работает.

Так же рекомендую установить параметр в zabbix **Timeout** в максимальное значение **30**. Обработка почтового лога pflogsumm может длиться несколько секунд. По-умолчанию, zabbix ждет 3 секунды ответа скрипта. Если за это время pflogsumm не обработает лог, заббикс получит ошибку. Обязательно время ожидания надо увеличить, так как 3 секунды очень мало.

> Нужно это сделать и на агенте, и на сервере. Я иногда забываю отредактировать конфиг сервера, а потом долго разбираюсь, почему не работает. В явном виде вы нигде не увидите ошибки ожидания по таймауту. Вместо этого хост будет уходить в оффлайн во время проверки длинных метрик, а потом сразу же возвращаться.

Если у вас будут ошибки вот на эти конструкции:

```
MAILQ=$(which mailq)
PFLOGSUMM=$(which pflogsumm)
```

то укажите пути к программам явно сами, например вот так:

```
MAILQ="/usr/bin/mailq"
PFLOGSUMM="/usr/sbin/pflogsumm"
```

И проверьте имя лог файла:

```
MAILLOG="/var/log/maillog"
```

В разных системах он имеет разное имя. Проверить работу скрипта можно следующим образом:

```
# sudo -u zabbix /etc/zabbix/scripts/postfix.sh queue
# sudo -u zabbix /etc/zabbix/scripts/postfix.sh received
```

В выводе вы должны видеть цифры, либо 0. Если в выводе пусто, разбирайтесь, в чем проблема. Посмотрите содержимое кэш файла

/tmp/zabbix.postfix.cache. Если в pflogsumm есть какие-то ошибки, вы увидите вывод команды в файле. Если под пользователем zabbix никак не хочет работать, проверьте сначала под root, просто запуская команды:

```
/etc/zabbix/scripts/postfix.sh queue
/etc/zabbix/scripts/postfix.sh received
```

После того, как убедитесь, что все работает, добавляем новые метрики в zabbix-agent. Для этого создаем файл с userparameters.

```
# mcedit /etc/zabbix/zabbix_agentd.d/userparameter_postfix.conf
```

```
UserParameter=postfix.received-full,pflogsumm -h 0 -u 0 --bounce_detail=0 --deferral_detail=0 --reject_detail=0 --
smtpd_warning_detail=0 --no_no_msg_size /var/log/maillog | grep received | awk 'NF == 2' | awk '{print $1}'
UserParameter=postfix.delivered-full,pflogsumm -h 0 -u 0 --bounce_detail=0 --deferral_detail=0 --reject_detail=0 --
smtpd_warning_detail=0 --no_no_msg_size /var/log/maillog | grep delivered | awk 'NF == 2' | awk '{print $1}'
UserParameter=postfix[*],/etc/zabbix/scripts/postfix.sh "$1"
```

Первые 2 ключа это просто накапливающиеся в течении всего дня значения полученных и доставленных писем. Значения получаю напрямую, выполняя подходящий запрос. Эти запросы будут актуальны в таком виде, только если вы настроите ротацию почтового лога раз в сутки в полночь. Если это не так, то используйте ключ **-d today**. Третий параметр собирает раз в 30 минут метрики от скрипта, показывая значения за последние 30 минут.

Перезапускайте агент и проверяйте, как он отдает значения.

```
# zabbix_agentd -t postfix[rejected]
 postfix[rejected] [t|3]
 # zabbix_agentd -t postfix.delivered-full
 postfix.delivered-all [t|15528]
 # zabbix_agentd -t postfix.received-full
```

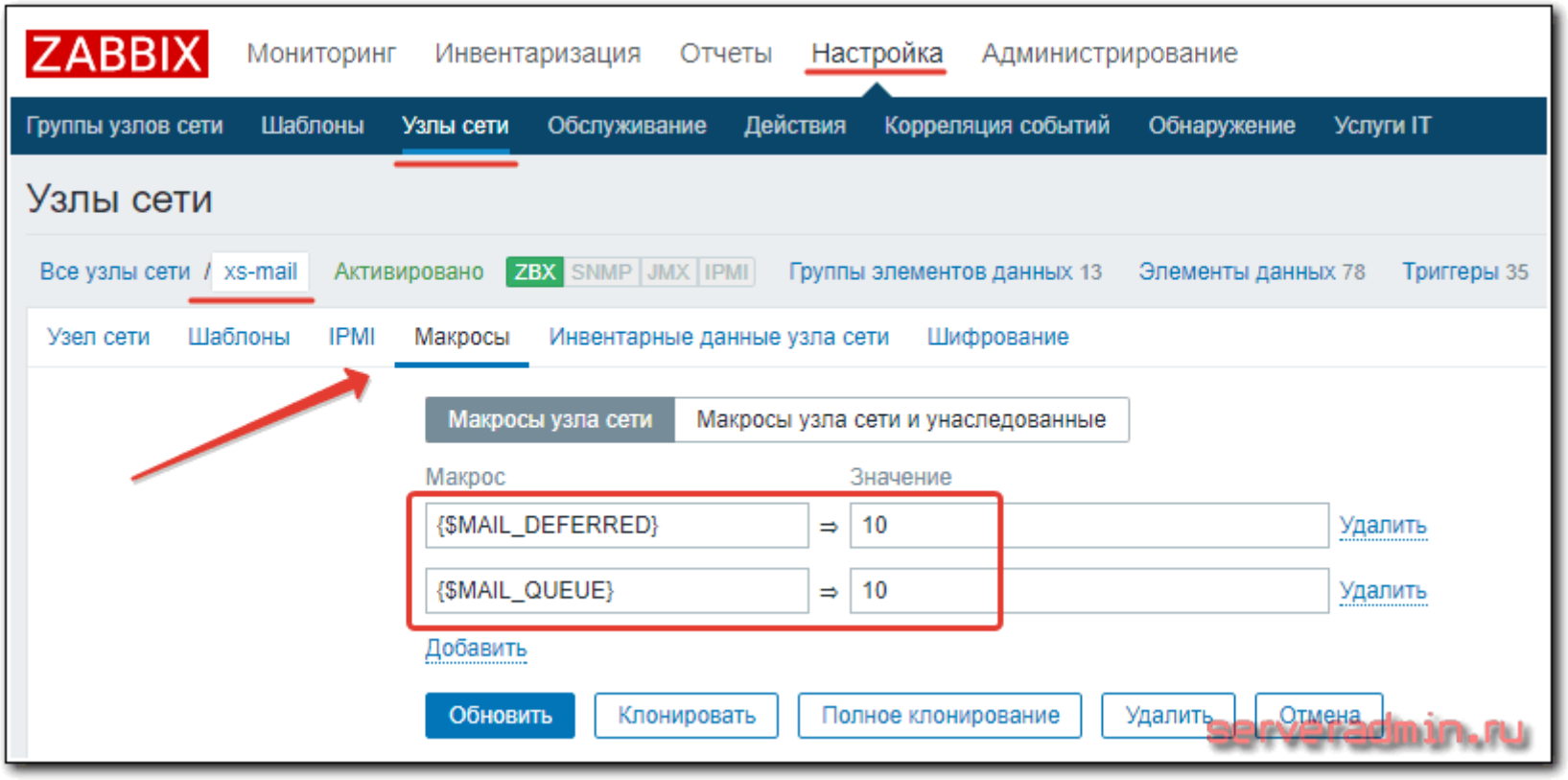
```
postfix.received-all [t|9511]
# zabbix_agentd -t postfix[queue]
postfix[queue] [t|0]
# zabbix_agentd -t postfix[rejected]
postfix[rejected] [t|3]
```

Только после того, как убедитесь, что все значения корректно возвращаются, можно переходить к настройке на zabbix сервере.

Настройка мониторинга postfix на zabbix сервере

Тут вам ничего настраивать не придется, так как я уже сделал готовый шаблон, который просто нужно будет импортировать и назначить нужным хостам. Вот сам шаблон - `template_postfix.xml`

Расскажу подробнее об элементах данных и триггерах в этом шаблоне. Во-первых, в шаблоне в триггерах используются макросы для определения максимальной длины очереди и отклоненных писем. После превышения, указанных в макросе значений, высылаются уведомления. Задать значения макросов можно либо в самом шаблоне, тогда значения для всех хостов будут одинаковые, либо в каждом отдельном хосте указываете свои значения. Я советую выставлять значения макросов на каждом хосте отдельно, подбирая оптимальное значения для каждого сервера. Делается это в свойствах хоста, в разделе Макросы.



В моем примере я установил значения макросов {$MAIL_DEFERRED} и {$MAIL_QUEUE} равными 10. Эти значения подобрал экспериментальным путем после нескольких дней работы мониторинга. Вообще, настроить сразу мониторинг почтового сервера не получится. Он требует длительной калибровки. Везде будут разные средние и максимально допустимые значения.

Вот список элементов данных в шаблоне:

Мастер	Имя	Триггеры	Ключ ▲	Интервал	История	Динамика изменений	Тип
	Postfix: Service imap	Триггеры 1	net.tcp.service[imap]	1м	7д	365д	Zabbix агент
	Postfix: Service smtp	Триггеры 1	net.tcp.service[smtp]	1м	7д	365д	Zabbix агент
	Postfix: Delivered Day		postfix.delivered-all	10м	90д	365д	Zabbix агент
	Postfix: Received Day		postfix.received-all	10м	90д	365д	Zabbix агент
	Postfix: Mails Bounced		postfix[bounced]	30м	7д	365д	Zabbix агент
	Postfix: bytes delivered		postfix[bytes delivered]	30м	7д	365д	Zabbix агент
	Postfix: bytes received		postfix[bytes received]	30м	7д	365д	Zabbix агент
	Postfix: Mails Deferred	Триггеры 1	postfix[deferred]	30м	7д	365д	Zabbix агент
	Postfix: Mails Delivered	Триггеры 1	postfix[delivered]	30м	7д	365д	Zabbix агент
	Postfix: Mails Discarded		postfix[discarded]	30м	7д	365д	Zabbix агент
	Postfix: Mails Forwarded		postfix[forwarded]	30м	7д	365д	Zabbix агент
	Postfix: Mails Held		postfix[held]	30м	7д	365д	Zabbix агент
	Postfix: QUEUE	Триггеры 3	postfix[queue]	1м	7д	365д	Zabbix агент
	Postfix: Mails Received		postfix[received]	30м	7д	365д	Zabbix агент
	Postfix: Mails Recipient Hosts		postfix[recipient hosts]	30м	7д	365д	Zabbix агент
	Postfix: Mails Recipients		postfix[recipients]	30м	7д	365д	Zabbix агент
	Postfix: Mails Rejected		postfix[rejected]	30м	7д	365д	Zabbix агент
	Postfix: Mails Sending Hosts		postfix[sending hosts]	30м	7д	365д	Zabbix агент
	Postfix: Number of processes master	Триггеры 1	proc.num[master]	1м	7д	365д	Zabbix агент
	Postfix: Number of processes pickup	Триггеры 1	proc.num[pickup]	1м	7д	365д	Zabbix агент
	Postfix: Number of processes qmgr	Триггеры 1	proc.num[qmgr]	1м	7д	365д	Zabbix агент

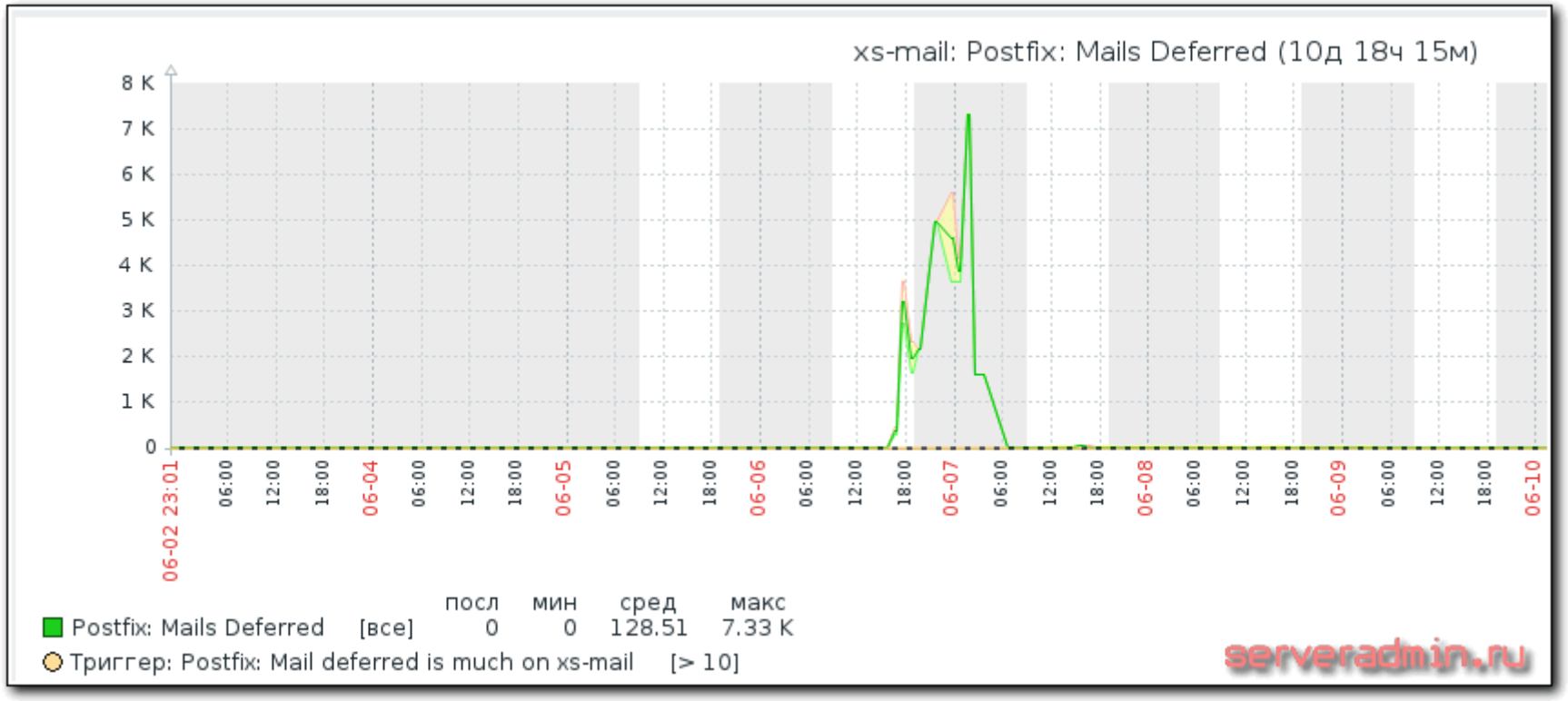
Список настроенных триггеров:

Важность	Имя ▲	Выражение
Чрезвычайная	Postfix: IMAP service is unreachable on {HOST.NAME} **Зависит от:** Template Postfix: Postfix: master is not running on {HOST.NAME}	{Template Postfix:net.tcp.service[imap].**last**(0)}<1
Предупреждение	Postfix: Mail deferred is much on {HOST.NAME}	{Template Postfix:postfix[deferred].**last**()}>{$MAIL_DEFERRED}
Средняя	Postfix: Mail delivered is much on {HOST.NAME}	Проблема: {Template Postfix:postfix[delivered].**sum**(3605)}>{Template Postfix:postfix[delivered].**sum**(3605,3605)}*5 Восстановление: {Template Postfix:postfix[delivered].**sum**(3605)}<{Template Postfix:postfix[delivered].**sum**(3605,3606)}*5
Предупреждение	Postfix: Mail queue is much on {HOST.NAME} **Зависит от:** Template Postfix: Postfix: Mail queue is too much on {HOST.NAME}	{Template Postfix:postfix[queue].**min**(180)}>{$MAIL_QUEUE}
Средняя	Postfix: Mail queue is too much on {HOST.NAME} **Зависит от:** Template Postfix: Postfix: Mail queue is very much on {HOST.NAME}	{Template Postfix:postfix[queue].**min**(180)}>{$MAIL_QUEUE}*1.5
Высокая	Postfix: Mail queue is very much on {HOST.NAME}	{Template Postfix:postfix[queue].**min**(180)}>{$MAIL_QUEUE}*2
Чрезвычайная	Postfix: master is not running on {HOST.NAME}	{Template Postfix:proc.num[master].**last**(0)}<1
Чрезвычайная	Postfix: pickup is not running on {HOST.NAME}	Проблема: {Template Postfix:proc.num[pickup].**count**(#3,1,"ne")}=3 Восстановление: {Template Postfix:proc.num[pickup].**count**(#3,1)}=3
Чрезвычайная	Postfix: qmgr is not running on {HOST.NAME}	{Template Postfix:proc.num[qmgr].**last**(0)}<1
Чрезвычайная	Postfix: SMTP service is unreachable on {HOST.NAME} **Зависит от:** Template Postfix: Postfix: master is not running on {HOST.NAME}	{Template Postfix:net.tcp.service[smtp].**last**(0)}<1

Расскажу подробнее об активных триггерах. Этот список состоит из всего полезного, что я нашел в интернете, плюс несколько моих триггеров. В существующих шаблонах, что мне попадались, я исправил некоторые условия, так как в них были ошибки и ложные срабатывания. Получилась моя собственная калькуляция, которую я некоторое время тестировал на живом активном сервере, прежде чем написать статью и поделиться с вами.

Триггеры проверки работы сервисов:

- **Postfix: IMAP service is unreachable on {HOST.NAME}** - стандартная проверка работы сервиса на определенном tcp порту. В данном случае это imap сервис и 143 порт.
- **Postfix: SMTP service is unreachable on {HOST.NAME}** - то же самое, только smtp порт 25.
- **Postfix: Mail queue is much on {HOST.NAME}** - проверка количества писем в почтовой очереди. Если минимальное значение за последние 3 минуты выше заданного в макросе $MAIL_QUEUE, то срабатывает триггер. Сам элемент данных проверяется каждую минуту. Сделаны 3 разных триггера с разными порогами срабатывания. В принципе, тут может быть достаточно и одного триггера, но я для примера оставил все три.
- **Postfix: Mail deferred is much on {HOST.NAME}** - триггер срабатывает, если количество отклоненных писем превышает значение, установленное в макросе $MAIL_DEFERRED. Это значение подбирается опытным путем. Проверка отклоненных писем и почтовой очереди самые эффективные для определения массовой рассылки спам писем с вашего сервера. Это может случиться, если пароль одного из ящиков уйдет к злоумышленникам. Ситуация эта достаточно частая. Во время отладки этого мониторинга, я как раз столкнулся с такой ситуацией и убедился, что рост значения deferred однозначно указывает на наличие какой-то проблемы. Когда вас взломают, вы получите примерно такую картинку.



Администрато

р указал при создании тестового ящика пароль 123456 и ночью его подобрали. Утром я встал и сразу все понял, просто посмотрев мониторинг.

- **Postfix: pickup is not running on {HOST.NAME}** - данный триггер контроллирует стандартный элемент данных, который получает информацию о работающем процессе на линукс сервере. Везде, где я видел шаблоны для мониторинга postfix, использовалась стандартная проверка работы сервиса - 0 или 1 в последнем полученном значении. Опытным путем я выяснил, что такая проверка дает много ложных срабатываний со службой pickup. Она запускается и работает ограниченное кол-во времени, потом завершается и запускается новый экземпляр. В мониторинг время от времени попадает информация о том, что сервис не работает, но в этом нет ошибки. Я сделал проверку трех последних состояний сервиса, и только в том случае, если все 3 проверки выявили неработающую службу, то будет оповещение. Остальные службы мониторятся просто последним значением. Они всегда должны быть запущены.
- **Postfix: Mail delivered is much on {HOST.NAME}** - этот триггер следит за количеством доставленных писем за последний час. Если их будет в 5

раз больше, чем часом ранее, то сработает оповещение. Этот триггер помогает следить за тем, что происходит на сервере. Если писем стало слишком много, стоит обратить на это внимание. Значение в 5 раз подобрано опытным путем. Возможно, в вашем случае будет уместно использовать другое значение. Тут только нужно понимать, что это за письма. Pflogsumm просто анализирует лог файл postfix. Значение delivered может расти, к примеру, если будут массовые рассылки. Кто-то сделает пару рассылок на весь офис, в котором 100 человек и вы сразу получите значение delivered 200. Если в предыдущем часе массовых рассылок не было, то значение delivered может быть в районе 20-30. Сразу получите сработавший триггер, который по сути не указывает на какую-то проблему. Так что калибруйте этот параметр в зависимости от ваших условий. В принципе, триггеры на queue и deferred достаточно четко фиксируют проблему на сервере, так что от этого триггера можно и отказаться. Привожу для примера, сами подумайте, как его использовать. Возможно, удобнее было бы использовать значение received для триггера, либо использовать и то и другое.

Описал ключевые значения мониторинга, на которые стоит реагировать. Возможно, я что-то упустил или сделал лишнее. Буду рад любым полезным и информативным замечаниям в комментариях. Вместе мы сможем более эффективно настроить мониторинг postfix. Может кто-то знает более информативные и удобные шаблоны, которые просто мне не попадались.

При написании триггеров, использовал описание функций триггеров из оригинальной документации.

Заключение

Мониторинг почтового сервера postfix нетривиальная и ответственная задача. Почтовые сервера постоянно находятся в зоне повышенного внимания со стороны злоумышленников, которые непрерывно пытаются найти уязвимые места и использовать ваш сервер для рассылки спама.

Ко всему прочему, сервис почтовых сообщений является критически важным сервисом, на который завязана работа многих организаций. Все это накладывает особую ответственность на администраторов почтовых серверов. Без хорошего мониторинга тут никак не обойтись.

Надеюсь, моя статья поможет вам повысить эффективность администрирования почтового сервера. Буду рад замечаниям, советам и информации о собственном опыте в мониторинге почты, в частности postfix. Хотя опыт других серверов тут тоже применим. Zabbix универсальный инструмент, на котором можно реализовать практически любые проверки.