ДОКУМЕНТАЦИЯ
по приложению JarvisX

ОГЛАВЛЕНИЕ

Версия документа 3
Типовая схема JarvisX 3
Сервер JarvisX 3
Список используемого ПО 4
Минимально требуемые ресурсы 4
Дисковая подсистема 5
Этапы запуска системы 5
- Первый запуск 5
- Запуск Nginx и приложений на php 6
  - Подготовка диска для nginx и приложений на php 6
  - Настройка Nginx 7
- Запуск приложения на Tomcat 13
  - Подготовка диска для Tomcat 13
- Запуск базы данных MariaDB 14
  - Первый запуск 14
  - Резервное копирование базы 15
  - Архивирование бинарных логов MariaDB 18

Сервис обновления и управления приложения 19
- Управление через командную строку 20
- Управление через web интерфейс 20
- Необходимые в работе JarvisX файлы настроек 21
- Подкладывание файлов с помощью манифеста 23
- Сбор информации при сбоях в работе приложения JarvisX 24

Мониторинг приложения 25
- Мониторинг основных показателей системы через Zabbix 25
- Мониторинг выполнения резервного копирования 26
- Настройка JMX для мониторинга загруженности ресурсов java 26
- Мониторинг загруженности ресурсов хоста через web 27

Версия документа

Контроль версий документа

Версия	Дата выпуска	Описание изменений	Автор (-ы)
1.0	12.01.2023	Проведена ревизия документа, внесены минорные правки	Синиченко Сергей
0.9	12.01.2023	Сформирована структура документа, внесена информация	Елизаров Кирил

Типовая схема JarvisX

Типовая схема JarvisX состоит из следующих компонент:

- фронтенд кеширующий сервер (Nginx) с возможностью загрузки статики без участия Tomcat;
- контейнер приложения JarvisX (java Tomcat);
- приложение СУБД (MariaDB);
- контейнер для приложений на php (phpMyAdmin).

Все компоненты собраны на одном виртуальном или физическом сервере (схема 1).

*Схема SEQ Drawing * ARABIC 1: Сервер JarvisX*

Сервер JarvisX

JarvisX сервер поставляется в виде образа диска виртуальной машины на FreeBSD 13, собранного с помощью NanoBSD скрипта. Логи, данные приложений и база данных размещаются на дополнительных дисках. Образ уже содержит необходимое для работы ПО и его настройки.

Образ диска ОС возможен в формате VHD (Hyper-V), VMDK (VMware) или сыром виде для дальнейшей конвертации. Образ содержит диск с 4 партициями:

1. Загрузчик GPT

2. Образ основной ОС с необходимым ПО, работающей в режиме ReadOnly.

3. Образ резервной ОС с необходимым ПО, работающей в режиме ReadOnly.

4. Область сохраненных конфигурационных данных ОС

Сервер рекомендуется размещать в DMZ сегменте сети. Желательно использовать на границе

между DMZ и Интернетом дополнительный балансировщик.

Список используемого ПО

Программное обеспечение	Версия	Ссылка

Операционная система FreeBSD	13.1	https://freebsd.org/
База данных MariaDB	10.4	https://mariadb.org/
Java среда разработки OpenJDK 11	11.0	https://github.com/battleblow/jdk11u/
Гипертекстовый процессор PHP	7.4	https://www.php.net/

Минимально требуемые ресурсы

Параметр	Значение
ЦПУ	8 процессоров на частоте 2.4 ГГц
ОЗУ	20 Гб
Сетевой интерфейс	1 Гбит/с
Диск ОС	16 Гб (SAS)
Диск базы данных	32 Гб (SSD)
Диск бинарных логов базы данных	32 Гб (SSD)
Диск приложений на php	32 Гб (SAS)
Диск приложений на Tomcat	48 Гб (SAS)

Также, дополнительно, потребуется диск для хранения резервных копий, обычно в качестве такого диска выступает NFS сервер.

Дисковая подсистема

Для работы используется штатная файловая система UFS, под диски приложений можно использовать ZFS. Диски приложений могут быть шифрованными.

Этапы запуска системы

Первый запуск

Создаем машину srvjx01: 8vCPU, 20GB RAM, 1 Ethernet (em1000, vmx).

Запускаем машину и заходим в консоль. Диск с ОС необходимо подключить первым. Далее последовательно надо добавить диски: базы данных, бинарных логов, Tomcat и php.

При первом запуске будет работать только ОС, все сервисы и контейнеры не будут запущены.

Настраиваем /etc/rc.conf, указав правильное имя хоста, его IP адрес и адрес шлюза

```
# cd /etc
# ee rc.conf
```

Редактируем строки:

```
hostname="srvjx01.somedomain.ru"                                 # имя хоста
defaultrouter="99.99.99.1"                                       # шлюз по-умолчанию
ifconfig_vmx0="inet 192.168.0.200 netmask 255.255.255.0"  # IP адрес и маска внутренней сети
```

Также настраиваем /etc/pf.conf файрвола, указав IP адрес хоста

```
# cd /etc
# ee pf.conf
```

Редактируем строку:

```
inside_ip="192.168.0.200"                    # IP адрес хоста
inside_mask="255.255.255.0"                  # маска сети
inside_gw="192.168.0.1"                      # шлюз сети
```

Проверяем правильность изменений файрвола

```
# pfctl -nf /etc/pf.conf                     # только проверяем синтаксис, правила не применяем
# pfctl -f /etc/pf.conf                      # применяем и перезапускаем файрвол
```

Сохраняем изменения в специальной партиции

```
# cd /etc/root
# sh sync_cfg
```

Перезапускаем ОС, после перезагрузки должен появиться доступ по SSH.

```
# init 6
```

Запуск Nginx и приложений на php

Подготовка диска для nginx и приложений на php

Дополнительный диск da3 предназначен для хранения кеша и логов nginx и приложений на php. Проверяем его наличие:

```
# dmesg | grep da3
da3 at mpt0 bus 0 scbus2 target 3 lun 0
da3: <VMware Virtual disk 2.0> Fixed Direct Access SPC-4 SCSI device
da3: 300.000MB/s transfers
da3: Command Queueing enabled
da3: 32768MB (67108864 512 byte sectors)
da3: quirks=0x140<RETRY_BUSY,STRICT_UNMAP>
```

Создаем структуру GPT на диске

```
# gpart create -s gpt da3
```

Создаем партицию

```
# gpart add -t freebsd-ufs da3
```

Форматируем партицию

```
# newfs /dev/da3p1
```

Открываем для редактирования /etc/fstab

```
# cd /etc
# ee fstab
```

И убираем символ комментирования со строчки:

```
#/dev/da3p1 ...
```

Монтируем диск

```
# mount /usr/local/www
```

Настройка Nginx

В папке /usr/local/nginx/ размещен основной конфигурационный файл Nginx.

Содержимое файла /usr/local/nginx/nginx.conf:

```
user  www;
worker_processes  2;

error_log  /usr/local/www/logs/nginx/error.log;

pid                     /var/run/nginx.pid;

worker_rlimit_nofile  200000;
events {
        worker_connections  50000;
        use kqueue;
}

http {
        charset                                         utf-8;
        server_tokens                                   off;
        include                                         mime.types;
        default_type                                    application/octet-stream;

    server_names_hash_max_size          256;
    server_names_hash_bucket_size       256;

        add_header                                      Strict-Transport-Security "max-age=31536000; includeSubdomains; preload";
        add_header                                      X-Frame-Options SAMEORIGIN;
        add_header                                      X-Content-Type-Options nosniff;
        add_header                                      X-XSS-Protection "1; mode=block";
        add_header                                      Content-Security-Policy "default-src 'self'; font-src 'self' data:; style-src *
'unsafe-inline'; script-src * 'unsafe-inline' 'unsafe-eval'; img-src * data: 'unsafe-inline'; connect-src * 'unsafe-inline'; frame-src *;";

        log_format main                                 '$remote_addr - $remote_user [$time_local] "$request" '
                                                        '$status $body_bytes_sent "$http_referer" '
                                                        '"$http_user_agent" "$http_x_forwarded_for" '
                                                        '$upstream_addr $request_time $upstream_connect_time '
                                                        '$upstream_header_time $upstream_response_time $pipe';

        access_log  /usr/local/www/logs/nginx/access.log  main;

        sendfile                                on;
        tcp_nopush                                      on;
        tcp_nodelay                                     on;

        keepalive_timeout                       5;
        client_body_timeout                     60;
        client_header_timeout                           60;
        send_timeout                                    60;

        gzip                                            on;
        gzip_min_length                                 1100;
        gzip_comp_level                                 5;
        gzip_http_version                       1.0;
        gzip_proxied                                    any;
        gzip_buffers                                    16 8k;
        gzip_types text/plain text/css application/x-javascript text/xml application/xml application/xml+rss text/javascript;
        gzip_disable                                    "MSIE [1-6].(?!.*SV1)";
        gzip_vary                                       on;

        limit_conn_zone $binary_remote_addr zone=perip:10m;

        proxy_cache_path /usr/local/www/nginx_cache levels=1 keys_zone=jx_cache:32m max_size=1g inactive=10m
use_temp_path=off;

    open_file_cache             max=200000 inactive=20s;
    open_file_cache_valid         30s;
    open_file_cache_min_uses        10;
    open_file_cache_errors        on;

    ssl_session_cache           shared:SSL:200m;
    ssl_session_timeout         30m;
```

```
        server {
                limit_conn perip                3;
                listen                          *:80 default accept_filter=httpready;
                server_name                     _;
                return                          404;
                access_log                      /usr/local/www/logs/honeypot/nginx_access.log  main;
        }

        server {
                listen                          *:80;
                server_name                     status.localhost;
                keepalive_timeout               0;
                access_log                      off;
                allow                           127.0.0.1;
                deny                            all;

                location =/status {
                        stub_status             on;
                }

        }

        server {
                listen                          *:443 default accept_filter=dataready ssl http2;
                server_name                     _;

                add_header                      Strict-Transport-Security "max-age=31536000; includeSubdomains; preload";

                ssl_certificate                 /usr/local/etc/nginx/ssl/somedomain.ru/somedimain.ru.crt;
                ssl_certificate_key             /usr/local/etc/nginx/ssl/somedomain.ru/somedimain.ru.key;
                client_max_body_size            20m;
                ssl_session_timeout             5m;
                ssl_protocols                   TLSv1.3 TLSv1.2;
                ssl_ciphers
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_256_GCM_SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20
-POLY1305:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-SHA384:ECDH
E-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA;
        ssl_conf_command            Ciphersuites TLS_CHACHA20_POLY1305_SHA256;
        ssl_conf_command            Options ServerPreference,PrioritizeChaCha,NoRenegotiation,NoResumptionOnRenegotiation;
        ssl_ecdh_curve              secp521r1:secp384r1;
                ssl_prefer_server_ciphers   on;
                ssl_dhparam                 /usr/local/etc/nginx/ssl/dhparam.pem;

                return                      404;
                access_log                  /usr/local/www/logs/honeypotssl/nginx_access.log  main;
        }

        upstream netdata {
                server 127.0.0.1:19999;
                keepalive 64;
        }

        upstream hawtio {
                server 127.0.0.1:9090;
        }

    server {
        listen                  *:443;
        server_name             srvjx01.somedomain.ru;

        add_header              Strict-Transport-Security "max-age=31536000; includeSubdomains; preload";

        ssl_certificate         /usr/local/etc/nginx/ssl/somedomain.ru/somedimain.ru.crt;
        ssl_certificate_key     /usr/local/etc/nginx/ssl/somedomain.ru/somedimain.ru.key;
        client_max_body_size    20m;
        ssl_session_timeout     5m;
        ssl_protocols           TLSv1.3 TLSv1.2;
                ssl_ciphers
```

```
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_256_GCM_SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20
-POLY1305:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-SHA384:ECDH
E-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA;
                ssl_conf_command        Ciphersuites TLS_CHACHA20_POLY1305_SHA256;
                ssl_conf_command        Options
ServerPreference,PrioritizeChaCha,NoRenegotiation,NoResumptionOnRenegotiation;
                ssl_ecdh_curve                  secp521r1:secp384r1;
        ssl_prefer_server_ciphers       on;
        ssl_dhparam             /usr/local/etc/nginx/ssl/dhparam.pem;

        location ~ /\. {
                deny all;
        }

        location /server-status {
                deny all;
        }

                location /logs {
                root                    /usr/local/tomcat/;
                        autoindex               on;
                        autoindex_exact_size            off;
                        autoindex_format        html;
                        autoindex_localtime     on;
                allow                   10.0.0.0/8;
                        deny                    all;
                }

                location /service {
                        root /usr/local/www/;
                index index.php index.html index.htm;
                location ~ ^/service/(.+\.php)$ {
                root /usr/local/www/;
                try_files $uri =404;
                fastcgi_pass unix:/tmp/php7.somedomain.ru_tmp/php-fpm.sock;
                fastcgi_index index.php;
                fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
                include /usr/local/etc/nginx/fastcgi_params;
                }
                location ~* ^/service/(.+\.(jpg|jpeg|gif|css|png|js|ico|html|xml|txt))$ {
                root /usr/local/www/;
                }
                        allow                   10.0.0.0/8;
                                deny                    all;
                        }

                        location /phpmyadmin {
                                root /usr/local/www/;
                                index index.php index.html index.htm;
                                location ~ ^/phpmyadmin/(.+\.php)$ {
                                        root /usr/local/www/;
                                        try_files $uri =404;
                                        fastcgi_pass unix:/tmp/php7.somedomain.ru_tmp/php-fpm.sock;
                                        fastcgi_index index.php;
                                        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
                                        include /usr/local/etc/nginx/fastcgi_params;
                                }
                                location ~* ^/phpmyadmin/(.+\.(jpg|jpeg|gif|css|png|js|ico|html|xml|txt))$ {
                                        root /usr/local/www/;
                                }
                        allow                   10.0.0.0/8;
                                deny                    all;
                        }
                        location /phpMyAdmin {
                                rewrite ^/* /phpmyadmin last;
                        allow                   10.0.0.0/8;
                                deny                    all;
                        }
```

```
        location = /status {
                return 301 /status/;
        allow           10.0.0.0/8;
                deny            all;
        }

        location ~ /status/(?<ndpath>.*) {
              proxy_redirect                    off;
                proxy_set_header        Host $host;
                proxy_set_header        X-Forwarded-Host $host;
                proxy_set_header        X-Forwarded-Server $host;
                proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_http_version      1.1;
                proxy_pass_request_headers on;
                proxy_set_header        Connection "keep-alive";
                proxy_store                     off;
                proxy_pass                      http://netdata/$ndpath$is_args$args;
                gzip                            on;
                gzip_proxied                    any;
                gzip_types                      *;
        allow           10.0.0.0/8;
                deny            all;
        }

    location /hawtio/ {
                proxy_pass                      http://hawtio/hawtio/;
          proxy_set_header      Host $host;
          proxy_set_header      X-Forwarded-Host $host;
          proxy_set_header      X-Forwarded-Server $host;
          proxy_set_header      X-Forwarded-For $proxy_add_x_forwarded_for;
                allow           10.0.0.0/8;
                deny            all;
    }

    root            /usr/local/www/srvjx01.somedomain.ru;
    set $root_path      /usr/local/www/srvjx01.somedomain.ru;
    disable_symlinks        if_not_owner from=$root_path;
    index           index.html;
    access_log          /usr/local/www/logs/srvjx01.somedomain.ru/nginx_access.log  main;
        location / {
        allow           192.168.100.33;
                deny            all;
        }
  }

    include                             /usr/local/etc/nginx/conf.d/*.conf;
}
```

В папке /usr/local/nginx/conf.d размещен конфигурационный файл приложения на Tomcat. upstream - это сервер Tomcat, который размещен в контейнере с ip 127.0.0.1.

Для сайта настроено кеширование файлов на 5 минут, с возможностью скачивать статические файлы напрямую, без участия Tomcat. Ниже представлено содержимое конфигурационного файла jxdemo.somedomain.ru.conf:

```
    upstream jxdemo.somedomain.ru  {
        server                          127.0.0.1:8080;
    }

    server {
        limit_conn perip        10;
        listen                          *:80;
```

```
                server_name                     jxdemo.somedomain.ru;
                return                          301
https://jxdemo.somedomain.ru/PO.Insurance/ru.lois.web.POInsurance/POInsurance.html;
        }


        server {
                listen              *:443;
                server_name             jxdemo.somedomain.ru;

        ssl_certificate         /usr/local/etc/nginx/ssl/jxdemo.somedomain.ru/jxdemo.somedomain.ru.crt;
        ssl_certificate_key     /usr/local/etc/nginx/ssl/jxdemo.somedomain.ru/jxdemo.somedomain.ru.key;
        client_max_body_size    20m;
        ssl_session_timeout     5m;
        ssl_protocols           TLSv1.3 TLSv1.2;
                ssl_ciphers
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_256_GCM_SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20
-POLY1305:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-SHA384:ECDH
E-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA;
                ssl_conf_command        Ciphersuites TLS_CHACHA20_POLY1305_SHA256;
                ssl_conf_command        Options
ServerPreference,PrioritizeChaCha,NoRenegotiation,NoResumptionOnRenegotiation;
                ssl_ecdh_curve                  secp521r1:secp384r1;
        ssl_prefer_server_ciphers   on;
        ssl_dhparam             /usr/local/etc/nginx/ssl/dhparam.pem;

                access_log                      /usr/local/www/jxdemo.somedomain.ru/nginx_access.log  main;

                root                            /usr/local/tomcat/webapps;

                proxy_cache                     jx_cache;
                proxy_cache_methods             GET HEAD;
                proxy_cache_lock        on;
                proxy_cache_lock_timeout    10s;
                proxy_ignore_headers            Cache-Control Expires Set-Cookie;
                proxy_cache_valid 200 302   5m;
                proxy_cache_valid any           1m;
                proxy_set_header        Host $host;
                proxy_set_header        X-Forwarded-Proto $scheme;
                proxy_set_header        X-Real-IP $remote_addr;
                proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_connect_timeout           600;
                proxy_send_timeout      600;
                proxy_read_timeout      600;

                location ~ /\. {
                        deny all;
                }

                location /server-status {
                        deny all;
                }

                if ($http_user_agent ~
(YandexBot|Googlebot|bot|Bot|AhrefsBot|DotBot|oBot|bingbot|Slackbot|Slackbot-LinkExpanding|AcoiRobot) ) {
                        return 404;
                }

                location /robots.txt {
                        return 200 "User-Agent: * \nDisallow: /";
                }

                location ~ \.do$ {
                        proxy_pass              http://jxdemo.somedomain.ru;
                        add_header              Cache-Control 'no-store, no-cache, must-revalidate, proxy-revalidate,
max-age=0';
                        proxy_no_cache          1;
                }
```

```
                location ~ \.jsp$ {
                        proxy_pass              http://jxdemo.somedomain.ru;
                        add_header              Cache-Control 'no-store, no-cache, must-revalidate, proxy-revalidate,
max-age=0';
                        proxy_no_cache          1;
                }

        location ~ .*DownloadStoredContentsAction.* {
                        proxy_pass              http://jxdemo.somedomain.ru;
                        add_header              Cache-Control 'no-store, no-cache, must-revalidate, proxy-revalidate,
max-age=0';
                        proxy_no_cache          1;
                }

        location ~ .*GetRunningActions.* {
                        proxy_pass              http://jxdemo.somedomain.ru;
                        add_header              Cache-Control 'no-store, no-cache, must-revalidate, proxy-revalidate,
max-age=0';
                        proxy_no_cache          1;
                }

            location ~ .*POInsurance.html?.* {
                        proxy_pass              http://jxdemo.somedomain.ru;
                        add_header              Cache-Control 'no-store, no-cache, must-revalidate, proxy-revalidate,
max-age=0';
                        proxy_no_cache          1;
                }

                location @tomcat {
                        proxy_pass              http://jxdemo.somedomain.ru;
                        add_header Cache-Control 'no-store, no-cache, must-revalidate, proxy-revalidate, max-age=0';
                }

                location / {
                        expires                 60;
                        if ($uri = "/") {
                    return          301 https://$server_name/PO.Insurance/ru.lois.web.POInsurance/POInsurance.html;
                        }
                        try_files $uri $uri/ @tomcat;
        }
}
```

В папке /usr/local/nginx/ssl размещаем сертификаты сайтов.

В папке /usr/local/www создаем папку логов logs:

```
# mkdir /usr/local/www/logs
```

Далее создаем необходимый для работы набор папок логов.

Папка логов сервера Nginx, сюда пишутся ошибки всех сайтов и логи самого Nginx:

```
# mkdir /usr/local/www/logs/nginx

# mkdir /usr/local/www/logs/honeypot

# mkdir /usr/local/www/logs/honeypotssl
```

Папка логов honeypot и honeypotssl, сюда пишутся факты подключения к nginx по IP, без использования имени сайта. В соответствующих папках находятся логи http и https.

Папка(и) логов сата(ов) для хранения запросов к сайту(там):

```
# mkdir /usr/local/www/logs/jxdemo.somedomain.ru
```

Перед перезапуском Nginx для применения новых конфигураций проверяем синтаксис командой:

```
# nginx -t
```

Запускаем Nginx

```
# /usr/local/etc/rc.d/nginx start
```

Сохраняем изменения в специальной партиции

```
# cd /etc/root
# sh sync_cfg
```

Запуск приложения на Tomcat

Подготовка диска для Tomcat

Дополнительный диск da4 предназначен для хранения приложения и логов на java. Проверяем его наличие:

```
# dmesg | grep da4
da4 at mpt0 bus 0 scbus2 target 4 lun 0
da4: <VMware Virtual disk 2.0> Fixed Direct Access SPC-4 SCSI device
da4: 300.000MB/s transfers
da4: Command Queueing enabled
da4: 49152MB (100663296 512 byte sectors)
da4: quirks=0x140<RETRY_BUSY,STRICT_UNMAP>
```

Создаем структуру GPT на диске

```
# gpart create -s gpt da4
```

Создаем партицию

```
# gpart add -t freebsd-ufs da4
```

Форматируем партицию

```
# newfs /dev/da4p1
```

Открываем для редактирования /etc/fstab

```
# cd /etc
# ee fstab
```

И убираем символ комментирования со строчки:

```
#/dev/da4p1 ...
```

Монтируем диск

```
# mount /usr/local/tomcat
```

Запуск базы данных MariaDB

База данных приложения работает под управлением MariaDB. База и бин логи размещаются на дополнительных двух дисках.

Первый запуск

Открываем для редактирования /etc/fstab

```
# cd /etc
# ee fstab
```

И убираем символ комментирования со строчек:

```
#/dev/da1p1 ...
#/dev/da2p1 ...
#md ...
```

Также настраиваем /etc/rc.conf, указав правильное имя хоста, его IP адрес и адрес шлюза

```
# cd /etc
# ee rc.conf
```

Редактируем строку запуска MariaDB:

```
mysql_enable="YES"                      # разрешаем запуск MariaDB
```

Создаем файл настроек инициализации базы данных. Для в редакторе создаем файл

```
# cd /usr/local/scripts/mysql
# ee install_mysql.conf
```

со следующим содержимым:

```
mysqlcheck_pass="SomeSecret1"     # Пароль пользователя, с правом доступа к статистике
dumpacct_pass="SomeSecret2"       # Пароль пользователя, с правом выполнения бекапов
repacct_pass="SomeSecret3"        # Пароль пользователя, с правом выполнения репликации
tempadmin_pass="SomeSecret4"    # Пароль временного пользователя для работы скрипта
mysqlacct_pass="VerySecretPass"   # Пароль администратора

db1="127.0.0.1"                   # IP адрес первого сервера MariaDB
db2="127.0.0.1"                   # IP адрес второго сервера MariaDB при наличии репликации, иначе
указать от db1
# Список баз, которые необходимо создать, по-одной базе в строке
# имя базы | имя пользователя для базы | пароль пользователя бд | права доступа (указать имя)
db_list='
        jxdb|jxacct|somejxpassword|jxacct
'
```

Запускаем скрипт инициализации базы данных:

```
# cd /usr/local/scripts/mysql
# ./install_mysql.sh
```

В результате работы скрипта будут созданы партиции на дисках da1 и da2, создана на них файловая система, выполнена инициализация и запуск приложения MariaDB.

Сохраняем изменения в специальной партиции

```
# cd /etc/root
# sh sync_cfg
```

Резервное копирование базы

Резервное копирование выполняется скриптом, который запускается ежедневно по cron. Для авторизации используется ранее созданная запись dumpacct. Запись ведется в

папку /bkp, обычно это точка монтирования отдельного диска или NFS сервера. В папке создается следующая структура:

Скрипт резервного копирования MariaDB:

```
#!/bin/sh

site="$(hostname)"
name="$(basename ${0})"
lpath="$(dirname ${0})"
status_date="$(date +%s)"
bkp_date="$(date "+%d%m%y"-"%H"-"%M")"
lockfile="/tmp/${name}.lock"
tmplogfile="/tmp/${name}${bkp_date}.log"
nfs_share="/bkp"
bkp_path="${nfs_share}"
status="/tmp/stat/${site}.dump.stat"
logfile="${bkp_path}/log/${name}.log"
dbpath="${bkp_path}/db"
dblink="${bkp_path}/db/current"
dumphome="/usr/local/scripts/mysql"
mysqlbin="/usr/local/bin/mysql"
mysqldumpbin="/usr/local/bin/mysqldump"
keepdays="3"

if [ -e ${lpath}/${name}.conf ]; then
        . ${lpath}/${name}.conf
fi

mail_send="NO"
bkp_online="NO"
bkp_status="OK"

dbquery="show databases;"
dblist="$(echo ${dbquery} | HOME=${dumphome} ${mysqlbin} 2> /dev/null | tail +2 | grep -v '_schema$' | grep -v '^sys$')"

if [ "$#" -gt "0" ]; then
        case $1 in
        [Dd][Ii][Ss][Cc][Oo][Vv][Ee][Rr])
                for db in ${dblist}
                do
                        json="$json,"'{"{#DBNAME}":"'${db# }'"}'
                done
                echo '{"data":['${json#,}']}'
                exit 0
        ;;
        [Ss][Tt][Aa][Tt][Uu][Ss])
                ret="$(/bin/cat "${status}" | grep ";${2};${3};" | head -n 1 | cut -d ';' -f 4)"
                if [ "${ret}" = '' ]; then
                        ret="0"
                fi
                echo -n "${ret}"
                exit 0
        ;;
        [Dd][Aa][Tt][Ee])
                ret="$(/bin/cat "${status}" | grep ";${2};${3};" | head -n 1 | cut -d ';' -f 1)"
                if [ "${ret}" = '' ]; then
                        ret="20000101"
                fi
                echo -n "${ret}"
                exit 0
        ;;
        [Dd][Uu][Mm][Pp])
                shift
                dblist="$@"
                [ -e "${status}" ] && cp "${status}" "${status}.tmp"
        ;;
        esac
```

```
else
        echo -n " > "${status}.tmp"
fi

echo "Start at $(date)" > "${tmplogfile}"

if [ -e "${bkp_path}" ]; then
        bkp_online="YES"
        echo "${bkp_path} is online" >> "${tmplogfile}"
else
        echo "${bkp_path} is offline" >> "${tmplogfile}"
fi

if [ "$(echo "${dblist}" | grep -c "mysql")" = "0" ]; then
        echo "Problem retrieving db list" >> "${tmplogfile}"
        echo "Got: ${dblist}" >> "${tmplogfile}"
else
        echo "Databases: ${dblist}" >> "${tmplogfile}"
        bkp_online="YES"
fi

if [ ! -e "${dbpath}" ] || [ ! -e "${dblink}" ]; then
        echo "Paths ${dbpath} or ${dblink} not found" >> "${tmplogfile}"
        bkp_status="FAILED"
        mail_send="YES"
        bkp_online="NO"
fi

if [ "${bkp_online}" = "YES" ]; then
        for db in ${dblist}
        do
                echo "${db} dump starts at $(date)" >> "${tmplogfile}"
                if [ "${db}" = "mysql" ]; then
                        HOME=${dumphome} ${mysqldumpbin} --events ${db} 2>> "${tmplogfile}" | sed 's/^CHANGE MASTER TO
MASTER_LOG_FILE=/#CHANGE MASTER TO MASTER_LOG_FILE=/' > ${dbpath}/${db}.${bkp_date}.sql
                        res="$?"
                else
                        HOME=${dumphome} ${mysqldumpbin} --max-allowed-packet=64m --single-transaction  --master-data ${db}
2>> "${tmplogfile}" | sed 's/^CHANGE MASTER TO MASTER_LOG_FILE=/#CHANGE MASTER TO MASTER_LOG_FILE=/' >
${dbpath}/${db}.${bkp_date}.sql
                        res="$?"
                fi
                if [ "${res}" = "0" ]; then
                        echo "${db} dump ends at $(date) successfully" >> "${tmplogfile}"
                        bkp_status="OK"
                else
                        echo "${db} dump ends at $(date) with error ${res}" >> "${tmplogfile}"
                        bkp_status="FAILED"
                        mail_send="YES"
                fi

                if [ "${res}" = "0" ]; then
                        echo "Changing rigths ..." >> "${tmplogfile}"
                        chmod 600 ${dbpath}/${db}."${bkp_date}".sql >> "${tmplogfile}" 2>&1
                        res="$?"
                        if [ "${res}" = "0" ]; then
                                echo "Done" >> "${tmplogfile}"
                        else
                                echo "Failed with ${res}" >> "${tmplogfile}"
                                bkp_status="FAILED"
                                mail_send="YES"
                        fi
                        echo "Changing link ..." >> "${tmplogfile}"
                        ln -vf ${dbpath}/${db}."${bkp_date}".sql ${dblink}/${db}.sql >> "${tmplogfile}" 2>&1
                        res="$?"
                        if [ "${res}" = "0" ]; then
                                echo "Done" >> "${tmplogfile}"
                        else
                                echo "Failed with ${res}" >> "${tmplogfile}"
```

```
                                        bkp_status="FAILED"
                                        mail_send="YES"
                                fi
                                echo "Deleting old dumps ..." >> "${tmplogfile}"
                                find ${dbpath} -name ${db}'.*.sql' -type f -mtime +${keepdays} -exec rm '{}' \; >> "${tmplogfile}" 2>&1
                                res="$?"
                                if [ "${res}" = "0" ]; then
                                        echo "Done" >> "${tmplogfile}"
                                else
                                        echo "Failed with ${res}" >> "${tmplogfile}"
                                        bkp_status="FAILED"
                                        mail_send="YES"
                                fi
                        fi

                        dump_stat="0"
                        if [ "${bkp_status}" = "OK" ]; then
                                dump_stat="1"
                        fi
                        if [ "$(cat ${status}.tmp | grep -c ";${db};dump;")" -gt "0" ]; then
                                sed -i -e "s/.*;${db};dump;.*/${status_date};${db};dump;${dump_stat}/" "${status}.tmp" && rm
"${status}.tmp-e"
                        else
                                echo "${status_date};${db};dump;${dump_stat}" >> "${status}.tmp"
                        fi
                        if [ "$(cat ${status}.tmp | grep -c ";${db};gzip;")" -gt "0" ]; then
                                sed -i -e "s/.*;${db};gzip;.*/${status_date};${db};gzip;1/" "${status}.tmp" && rm "${status}.tmp-e"
                        else
                                echo "${status_date};${db};gzip;1" >> "${status}.tmp"
                        fi
                done
fi

echo "End at $(date)" >> "${tmplogfile}"

[ -e "${status}.tmp" ] && mv "${status}.tmp" "${status}" && chmod 644 "${status}"

if [ "${mail_send}" = "YES" ]; then
        cat "${tmplogfile}" | /usr/local/bin/email -s "${name} runtime result from $(/bin/hostname)" -r smtp.somedomain.ru
postmaster@somedomain.ru
fi

cat "${tmplogfile}" >> "${logfile}"
rm "${tmplogfile}"
```

Архивирование бинарных логов MariaDB

Скрипт запускается раз в полчаса, генерит через SQL запрос закрытие текущего бинарного лога и копирует вновь созданные с момента прошлого копирования бинарные логи в папку /bkp/mysql-bin.

```
#!/bin/sh

set -e

site="$(hostname)"
name="$(basename ${0})"
status_date="$(date +%Y%m%d)"
bkp_date="$(date "+%d%m%y"-"%H"-"%M")"
lockfile="/tmp/${name}.lock"
tmplogfile="/tmp/${name}${bkp_date}.log"
nfs_share="/bkp"
bkp_path="${nfs_share}"
logfile="${bkp_path}/log/${name}.log"
status="/tmp/stat/${site}.dump.stat"
```

```
local_bin="/var/log/my/mysql-bin"
nfs_bin="${bkp_path}/mysql-bin"

dumphome="/usr/local/scripts/mysql"
mysqlbin="/usr/local/bin/mysql"

if [ -e ${lpath}/${name}.conf ]; then
        . ${lpath}/${name}.conf
fi

mail_send="NO"
keep_log="NO"

echo "Start at $(date)" > "${tmplogfile}"

if [ -e "${bkp_path}" ]; then
        echo "${bkp_path} is online" >> "${tmplogfile}"
else
        echo "${bkp_path} is offline" >> "${tmplogfile}"
                mail_send="YES"
fi

local_bin_list="$(ls ${local_bin})"

echo "Flushing binary logs ..." >> "${tmplogfile}"
HOME=${dumphome} ${mysqlbin} -e 'flush binary logs;' >> "${tmplogfile}" 2>&1
res="$?"
if [ "${res}" = "0" ]; then
                echo "Done" >> "${tmplogfile}"
else
                echo "Failed with ${res}" >> "${tmplogfile}"
                mail_send="YES"
fi

for F in ${local_bin_list}
do
#               echo "Comparing ${F} ..." >> "${tmplogfile}"
                if [ "$(stat -f %z ${local_bin}/${F})" != "$(stat -f %z ${nfs_bin}/${F})" ]; then
                                keep_log="YES"
                                echo "Copying ${local_bin}/${F} to ${nfs_bin} ..." >> "${tmplogfile}"
                                cp -v ${local_bin}/${F} ${nfs_bin} >> "${tmplogfile}" 2>&1
                                res="$?"
                                if [ "${res}" = "0" ]; then
                                                echo "Done" >> "${tmplogfile}"
                                else
                                                echo "Failed with ${res}" >> "${tmplogfile}"
                                                mail_send="YES"
                                fi
                fi
done

for F in $(ls ${nfs_bin})
do
#               echo "Comparing ${F} ..." >> "${tmplogfile}"
                if [ ! -e "${local_bin}/${F}" ]; then
                                keep_log="YES"
                                echo "Deleting ${nfs_bin}/${F} ..." >> "${tmplogfile}"
                                rm ${nfs_bin}/${F} >> "${tmplogfile}" 2>&1
                                res="$?"
                                if [ "${res}" = "0" ]; then
                                                echo "Done" >> "${tmplogfile}"
                                else
                                                echo "Failed with ${res}" >> "${tmplogfile}"
                                                mail_send="YES"
                                fi
                fi
done

echo "End at $(date)" >> "${tmplogfile}"
```

```
if [ "${mail_send}" = "YES" ]; then
        cat "${tmplogfile}" | /usr/local/bin/email -s "${name} runtime result from $(/bin/hostname)" -r smtp.somedomain.ru
postmaster@somedomain.ru
fi

if [ "${keep_log}" = "YES" ]; then
                cat "${tmplogfile}" >> "${logfile}"
fi

rm "${tmplogfile}"
```

Сервис обновления и управления приложения

Управление приложением может осуществляться как через командную строку так и через web интерфейс. Сервис содержит следующие утилиты:

```
# ls /usr/local/scripts/jx
hawtio           # запуск сервиса мониторинга java
jx-heapdump  # снятие heap дампа java
jx-histogram   # получение гистограммы java
jx-manifest     # запуск манифеста
jx-restart        # рестарт приложения
jx-stack          # получение стека
jx-start          # запуск приложения
jx-status         # статус состояния процесса
jx-stop           # останов приложения
jx-upgrade      # запуск обновления приложения
jx.conf           # конфигурационный файл сервиса
jxprefs.java     # приложение на java для внесения данных в реестр
jxsync            # сам скрипт управления
tomcat            # скрипт управления томкатом
watchdog        # скрипт управления, который отслеживает запросы через web приложение
```

Управление через командную строку

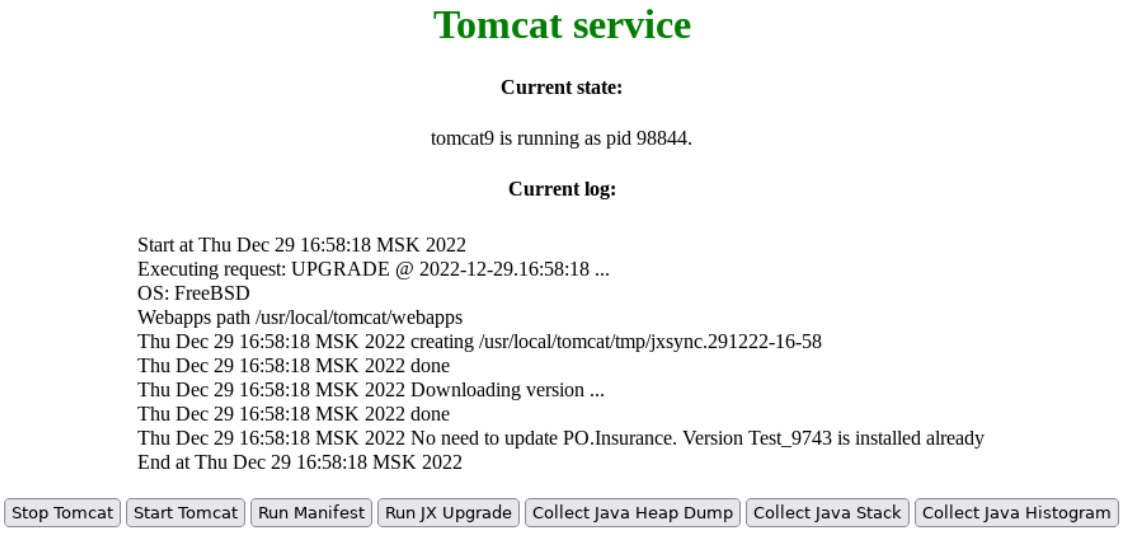
Управление из командной строки осуществляется стандартными вызовами скриптов в sh. Например так можно посмотреть текущий статус приложения:

```
# /usr/local/scripts/jx/jx-status
```

Управление через web интерфейс

Web интерфейс размещен по имени хоста, на котором размещено приложение, с указанием пути service, например: https://srvjx01.somedomain.ru/service.

Пример запуска web страницы управления:



В верхней части экрана указывается состояние приложения томкат. Например, в данном случае показано, что процесс tomcat9 работает и имеет id процесса 98844.

В средней части экрана располагается лог выполненной операции.

В нижней части экрана расположено меню:

Кнопка меню	Вызываемый скрипт	Описание
Stop Tomcat	/usr/local/scripts/jx/jx-stop	Остановить томкат
Start Tomcat	/usr/local/scripts/jx/jx-start	Запустить томкат
Run Manifest	/usr/local/scripts/jx/jx-manifest	Загрузить и исполнить файл манифеста
Run JX Upgrade	/usr/local/scripts/jx/jx-upgrade	Выполнить обновление приложения
Collect Java Heap Dump	/usr/local/scripts/jx/jx-heapdump	Собрать дамп памяти java приложения
Collect Java Stack	/usr/local/scripts/jx/jx-stack	Собрать текущий стек потоков java приложения
Collect Java Histogram	/usr/local/scripts/jx/jx-histogram	Собрать гистограмму объектов java приложения

Для выполнения операции нужно нажать кнопку и дождаться выполнения скрипта, результат выполненных работ отобразится через некоторое время в средней части экрана.

Обновление приложения происходит в установленное время программой автообновления путем скачивания архива с сайта обновлений с последующим разворачиванием архива на диск.

Если по правилам безопасности скачивание обновлений с ресурсов в Интернет запрещено, то можно развернуть локалный сайт с basic авторизацией для обновлений и указать его параметры в конфигурационном файле скрипта синхронизации.

В корне сайта создается папка имени сервера, например srvjx.somedomain.ru

Для выполнения обновления необходимо выложить на сайте version.xml с текущими версиями и архивы приложений в формате указанном выше. При запуске сервиса автообновления последний скачивает сначала настройки базы, затем файл с версиями, и, если видит, что версии другие, то скачивает архивы приложений. После чего распаковывает архивы, останавливает приложение и затем его запускает, java процесс затем выполняет необходимые обновления базы данных.

Необходимые в работе JarvisX файлы настроек

Необходимые в работе файлы настроек настраивает исполнитель, которые заказчик скачивает через обновление по манифесту. Ниже приводится перечень наиболее важных файлов.

- **partner_list.xml** - файл содержит некоторые настройки JarvisX, данный файл должен быть размещен до момента первичной инициализации базы данных.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PartnerList xmlns="http://www.lois.ru/">
      <partner>
      <partnerID>АО СК «Страхование»</partnerID>
            <defTopMarginOsago>22</defTopMarginOsago>
      <defLeftMarginOsago>6</defLeftMarginOsago>
      <vigrujatIPokazivatVOsagoProcentAV>false</vigrujatIPokazivatVOsagoProcentAV>
      <vigrujatIPokazivatVOsagoProdavecSK>false</vigrujatIPokazivatVOsagoProdavecSK>
      <vigrujatSozdatelDogovora>false</vigrujatSozdatelDogovora>
      <skrivatBannerOceni>false</skrivatBannerOceni>
      <defaultForEosago>false</defaultForEosago>
   </partner>
</PartnerList>
```

- **jxdb.txt** - файл настроек базы данных, пример содержимого файла настроек базы данных:

```
PO.InsuranceMAIL_TO=
PO.InsuranceOK=true
PO.InsuranceIDENTIFIKATOR=
PO.InsuranceHOST=localhost                    # адрес базы
PO.InsuranceIMA_BAZI=somedb                   # имя базы
PO.InsurancePOLZOVATEL=somedbuser             # пользователь с полными правами на базу
```

```
PO.InsurancePAROL=U29tZVBhc3N3b3Jk              # пароль для доступа в базу в формате base64
PO.InsurancePORT=1521
PO.InsuranceTABLESPACE=
PO.InsuranceDBTYPE=mysql                        # тип базы
PO.InsuranceKATALOG_TOMCAT=/usr/local/tomcat/tomcat.poinsurance    # путь до томката
```

- **version.xml** - файл с номерами текущих версий, приложение JarvisX скачивает его с сайта обновлений и обновляется, опираясь на его данные:

```
<distribut>
 <release>
  <name>PO.Insurance</name>
  <releaseversion>1.2.1</releaseversion>
  <buildversion>3698</buildversion>
 </release>
 <release>
  <name>JXConnector</name>
  <releaseversion>1.0.0</releaseversion>
  <buildversion>11</buildversion>
 </release>
</distribut>
```

Автозагрузчик скачивает версии приложений по номерам в version.xml. Например для приложения PO.Insurance по указанным выше данным будет скачиваться файл <ReleaseName>_<releaseversion>_<buildversion>.zip или PO.Insurance_1.2.1_3698.zip.

- **JXConnector_1.0.0_11.zip** - архив модуля приложения JarvisX
- **PO.Insurance_1.2.1_3698.zip** - архив модуля приожения JarvisX
- **server.xml** - содержит необходимые для работы Tomcat настройки

```
<?xml version="1.0" encoding="UTF-8"?>
<Server port="8005" shutdown="SHUTDOWN">
  <Listener className="org.apache.catalina.startup.VersionLoggerListener" />
  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
  <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
  <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />
  <GlobalNamingResources>
    <Resource name="UserDatabase" auth="Container"
              type="org.apache.catalina.UserDatabase"
              description="User database that can be updated and saved"
              factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
              pathname="conf/tomcat-users.xml" />
  </GlobalNamingResources>
  <Service name="Catalina">
      <Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000"
                 redirectPort="443" compression="on" useSendfile="false"

compressableMimeType="text/html,text/xml,text/plain,text/javascript,text/css,application/javascript,application/xls"/>
    <Engine name="Catalina" defaultHost="localhost">
      <Realm className="org.apache.catalina.realm.LockOutRealm">
        <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
```

```
            resourceName="UserDatabase"/>
      </Realm>
      <Host name="localhost"  appBase="webapps"
            unpackWARs="true" autoDeploy="true">
              <Valve className="org.apache.catalina.valves.ErrorReportValve" showReport="false" showServerInfo="false" />
        <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
               prefix="localhost_access_log" suffix=".txt"
               pattern="%h %l %u %t &amp;quot;%r&amp;quot; %s %b" />
        <Valve className="org.apache.catalina.valves.RemoteIpValve"
               protocolHeader="X-Forwarded-Proto" proxiesHeader="X-Forwarded-by" remoteIpHeader="X-Forwarded-For"
               internalProxies="127\.0\.0\.1" />
      </Host>
    </Engine>
  </Service>
</Server>
```

Необходимо настроить параметр internalProxies, указав правильный список ip адресов nginx. В нашем случае адрес 127.0.0.1, т. к. все сервисы находятся на одном хосте и связь между ними идет через loopback интерфейс.

- **variables.conf** – файл настроек java машины. Пример файла

```
# jx
-server
-Xmx8G
-XX:NewRatio=3
-XX:+CMSParallelRemarkEnabled
-XX:MaxMetaspaceSize=512M
-XX:+CMSScavengeBeforeRemark
-Xlog:gc*,safepoint:file=/usr/local/tomcat/gc/gc.log:time,uptime:filecount=100,filesize=128K
-XX:+UseCMSInitiatingOccupancyOnly
-Djava.net.preferIPv4Stack=true
-Djava.net.preferIPv4Addresses=true
-Dappendedfontpath=/usr/local/tomcat/fonts/
-Dcom.sun.xml.internal.bind.v2.runtime.JAXBContextImpl.fastBoot=true
-Dcom.sun.xml.bind.v2.runtime.JAXBContextImpl.fastBoot=true
-Dsun.net.inetaddr.ttl=60
-Dfile.encoding=UTF-8
#-XX:+UseConcMarkSweepGC
#-XX:ParallelGCThreads=2
#-XX:+ScavengeBeforeFullGC
#-XX:+DisableExplicitGC
-XX:+UseShenandoahGC
-XX:+UnlockExperimentalVMOptions
-XX:ShenandoahUncommitDelay=1000
-XX:ShenandoahGuaranteedGCInterval=10000
-Dcom.sun.management.jmxremote=true
-Dcom.sun.management.jmxremote.port=12345
-Dcom.sun.management.jmxremote.rmi.port=12346
-Dcom.sun.management.jmxremote.local.only=false
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
```

Подкладывание файлов с помощью манифеста

В папке обновлений сервера приложений необходимо создать текстовый файл manifest.txt, если его нет. Например для приложения srvjx.somedomain.ru содержимое папки выглядит так:

```
-rw-r--r--  1 root  jx_update       707  1 дек.   2021 jxdb.txt
-rw-r--r--  1 root  jx_update       191 30 марта 18:22 manifest.txt
-rw-r--r--  1 root  jx_update       575 22 марта 11:58 partner_list.xml
-rwxr--r--  1 root  jx_update  357624759 30 авг.  14:37 PO.Insurance_Test_8440.zip
-rwxr--r--  1 root  jx_update  357624795 30 авг.  16:02 PO.Insurance_Test_8442.zip
-rwxr--r--  1 root  jx_update    5242805 24 авг.  12:18 some-ui_Test_435.zip
-rwxr--r--  1 root  jx_update    5243616 24 авг.  18:13 some-ui_Test_437.zip
-rw-r--r--  1 root  jx_update       5130 30 марта 12:29 SendRastorg2RSA2Action.class
-rw-r--r--  1 root  jx_update       609 22 марта 13:42 variables.conf
-rwxr--r--  1 root  jx_update       332 31 авг.  10:37 version.xml
-rwxr--r--  1 root  jx_update       206  3 июня  17:46 version.xml.old
```

Далее необходимо открыть в редакторе файл manifest.txt. Ниже пример его содержимого:

```
#jxdb.txt|@ru/lois/web/installWeb/jxprefs.java
#partner_list.xml|conf
#variables.conf|conf
#SendRastorg2RSA2Action.class|webapps/PO.Insurance/WEB-INF/classes/ru/lois/web/actions
```

Строки, начинающиеся с #, считаются комментариями и не обрабатываются.
Для выкладки файла необходимо написать строку состоящую из имени файла, который необходимо загрузить, разделителя „|“ (вертикальная черта), и пути, по которому надо положить файл в приложении. Например для файла SendRastorg2RSA2Action.class, который надо положить по пути webapps/PO.Insurance/WEB-INF/classes/ru/lois/web/actions, строка манифеста будет выглядеть так:

```
SendRastorg2RSA2Action.class|webapps/PO.Insurance/WEB-INF/classes/ru/lois/web/actions
```

Папки должны отделяться прямым слешем «/».
Сам файл SendRastorg2RSA2Action.class необходимо предварительно скопировать в папку обновлений приложения, рядом с файлом манифеста.
Процесс обработки манифеста выполняется по команде

```
/usr/local/scripts/jx/jx-manifest
```

Или через веб сервис обновления нажатием кнопки «Run Manifest». По окончании обработки манифеста необходимо по логам проверить, что не было ошибок.

Сбор информации при сбоях в работе приложения JarvisX

При возникновении проблем в работе приложения можно, согласовав эти действия с поддержкой, попробовать его перезапустить командами:

```
# /usr/local/scripts/jx/tomcat stop
# /usr/local/scripts/jx/tomcat start
```

Или сделать аналогичный вызов через веб сервис обновлений.

Предварительно может потребоваться снять дампы приложения командами:

```
# /usr/local/scripts/jx/jx-heapdump
# /usr/local/scripts/jx/jx-stack
# /usr/local/scripts/jx/jx-histogram
```

Или вызвать эти команды через веб сервис обновлений.

Логи приложения и дампы можно скачать по ссылке https://srvjx01.somedomain.ru/logs

Мониторинг приложения

Для мониторинга ресурсов приложения есть несколько решений:

1. Мониторинг загруженности основных ресурсов через Zabbix
2. Мониторинг состояния java процесса через надстройку JMX для агента Zabbix
3. Мониторинг загруженности основных ресурсов через web сервис.

Мониторинг основных показателей системы через Zabbix

Конфигурационный файл агента находится в /usr/local/etc/zabbix5/zabbix_agentd.conf, его необходимо настроить под свой сервер забикс, указав IP сервера.
Шаблоны, необходимые для работы сервера zabbix, идут в комлекте поставки.

```
Server=10.*.*.10
ServerActive=10.*.*.10
HostnameItem=system.hostname
LogType=system
LogFileSize=0
DebugLevel=0
Timeout=12
HostMetadata=FreeBSD
Include=/usr/local/etc/zabbix5/zabbix_agentd.conf.d/*.conf
```

Мониторинг процессов mysql вынесен в отдельный файл /usr/local/etc/zabbix5/zabbix_agentd.conf.d/userparameter_mysql.conf

```
UserParameter=mysql.version,/usr/local/bin/mysql -V
UserParameter=mysql.status[*],( echo "show global status where Variable_name='$1';" |
HOME=/usr/local/etc/zabbix5 /usr/local/bin/mysql -N 2> /dev/null || echo "0 0" ) | awk '{print $$2}'
UserParameter=mysql.ping,HOME=/usr/local/etc/zabbix5 /usr/local/bin/mysqladmin ping 2> /dev/null |
/usr/bin/grep -c alive
UserParameter=mysql.size[*],echo "select sum($(case "$3" in both|"") echo "data_length+index_length";;
data|index) echo "$3_length";; free) echo "data_free";; esac)) from information_schema.tables$( ([ ! "$1" ] || [
```

```
"$1" = "all" ]) || echo " where table_schema='$1'")$( ([ "$2" = "all" ] || [ ! "$2" ]) || echo "and table_name='$2'");" |
HOME=/usr/local/etc/zabbix5 /usr/local/bin/mysql -N 2> /dev/null
UserParameter=mysql.srt, HOME=/usr/local/etc/zabbix5 /usr/local/bin/mysqladmin extended-status 2> /dev/null
| /usr/bin/grep -i 'Slave_retried_transactions' | cut -f3 -d'|'
UserParameter=mysql.sott,HOME=/usr/local/etc/zabbix5 /usr/local/bin/mysqladmin extended-status 2>
/dev/null | /usr/bin/grep -i 'Slave_open_temp_tables' | cut -f3 -d'|'
UserParameter=mysql.sr,  ( HOME=/usr/local/etc/zabbix5 /usr/local/bin/mysql -e 'show slave status\G' 2>
/dev/null || echo 'Slave_SQL_Running:No' ) | ( /usr/bin/grep  'Slave_SQL_Running:' || echo
'Slave_SQL_Running:No' ) | /usr/bin/cut -f2 -d: | tr -d '[:space:]' | sed "s/No/0/; s/Yes/1/"
UserParameter=mysql.sbm, ( HOME=/usr/local/etc/zabbix5 /usr/local/bin/mysql -e 'show slave status\G' 2>
/dev/null || echo 'Seconds_Behind_Master:999' ) | ( /usr/bin/grep  'Seconds_Behind_Master:' || echo
'Seconds_Behind_Master:999' ) | /usr/bin/cut -f2 -d: | tr -d '[:space:]' | sed 's/NULL/999/'
UserParameter=mysql.sle, ( HOME=/usr/local/etc/zabbix5 /usr/local/bin/mysql -e 'show slave status\G' 2>
/dev/null || echo 'Last_Errno:999' ) | ( /usr/bin/grep  'Last_Errno:' || echo 'Last_Errno:999' ) | /usr/bin/cut -f2 -d: |
tr -d '[:space:]' | sed 's/NULL/999/'
UserParameter=mysql.lioe, ( HOME=/usr/local/etc/zabbix5 /usr/local/bin/mysql -e 'show slave status\G' 2>
/dev/null || echo 'Last_IO_Errno:999' ) | ( /usr/bin/grep  'Last_IO_Errno:' || echo 'Last_IO_Errno:999' ) |
/usr/bin/cut -f2 -d: | tr -d '[:space:]' | sed 's/NULL/999/'
UserParameter=mysql.lse, ( HOME=/usr/local/etc/zabbix5 /usr/local/bin/mysql -e 'show slave status\G' 2>
/dev/null || echo 'Last_SQL_Errno:999' ) | ( /usr/bin/grep  'Last_SQL_Errno:' || echo 'Last_SQL_Errno:999' ) |
/usr/bin/cut -f2 -d: | tr -d '[:space:]' | sed 's/NULL/999/'
UserParameter=custom.backup.discover_databases[*],/usr/local/scripts/mysql/backup.sh discover
UserParameter=custom.backup.dump.status[*],/usr/local/scripts/mysql/backup.sh status $1 dump
UserParameter=custom.backup.dump.date[*],/usr/local/scripts/mysql/backup.sh date $1 dump
UserParameter=custom.backup.gzip.status[*],/usr/local/scripts/mysql/backup.sh status $1 gzip
UserParameter=custom.backup.gzip.date[*],/usr/local/scripts/mysql/backup.sh date $1 gzip
```

Мониторинг выполнения резервного копирования

Результат выполнения резервного копирования можно также мониторить через Zabbix. Для этого агенту можно добавить следующие параметры (идут в комплекте):

```
UserParameter=custom.backup.discover_databases[*],/usr/local/scripts/mysql/backup.sh discover
UserParameter=custom.backup.dump.status[*],/usr/local/scripts/mysql/backup.sh status $1 dump
UserParameter=custom.backup.dump.date[*],/usr/local/scripts/mysql/backup.sh date $1 dump
UserParameter=custom.backup.gzip.status[*],/usr/local/scripts/mysql/backup.sh status $1 gzip
UserParameter=custom.backup.gzip.date[*],/usr/local/scripts/mysql/backup.sh date $1 gzip
```

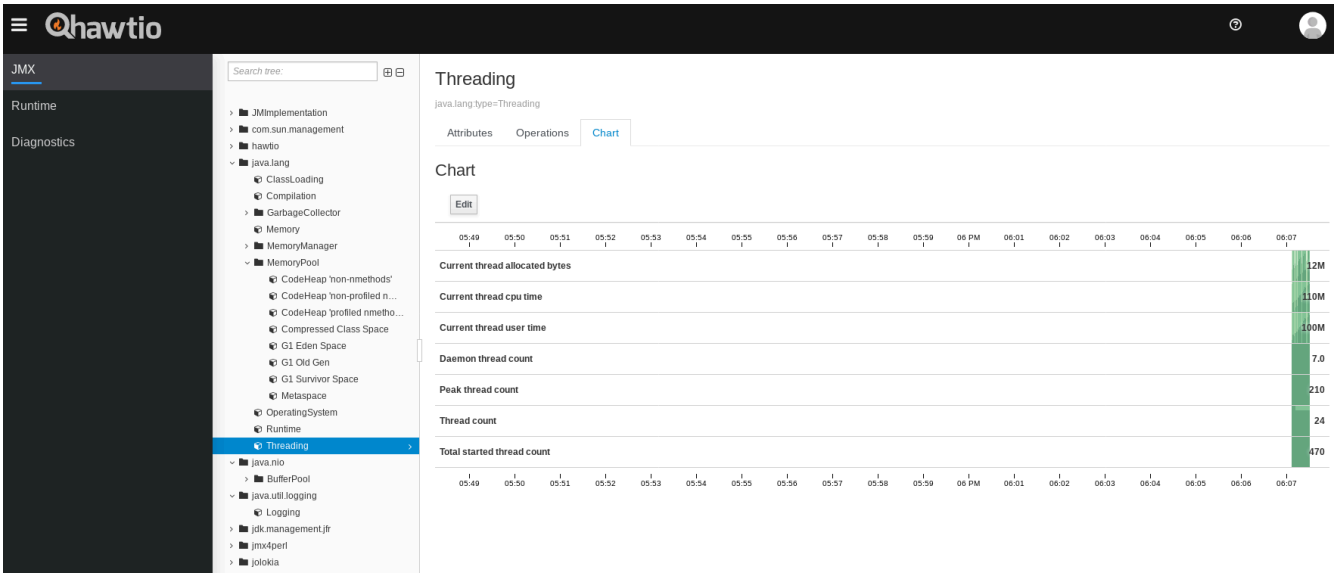
А на сервер забикс темплейт, идущий в комлекте поставки.

Настройка JMX для мониторинга загруженности ресурсов java

JMX настроен на стороне java приложения и готов отвечать на запросы:

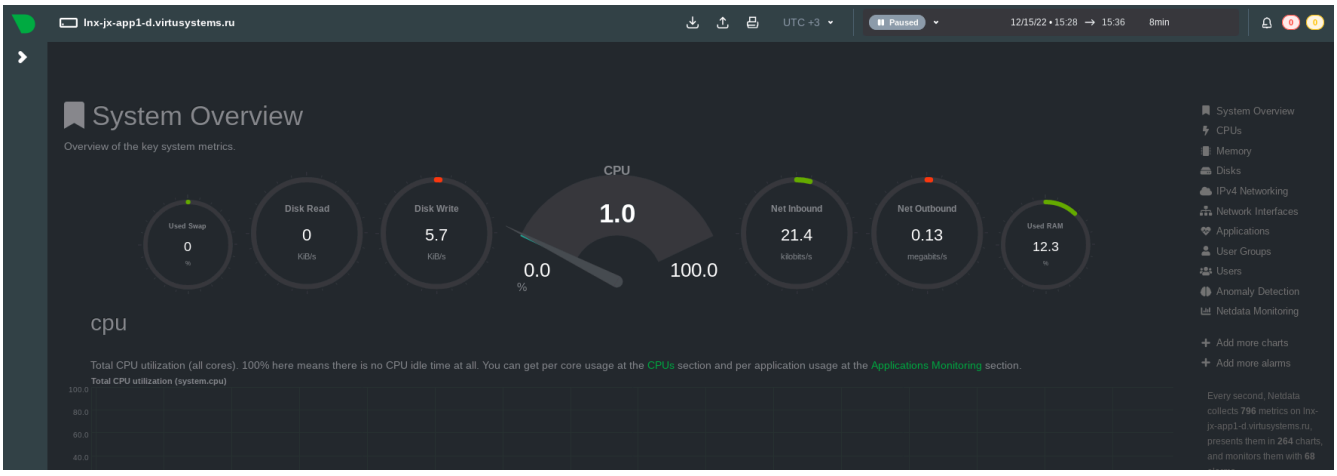
```
-Dcom.sun.management.jmxremote=true
-Dcom.sun.management.jmxremote.port=12345
-Dcom.sun.management.jmxremote.rmi.port=12346
-Dcom.sun.management.jmxremote.local.only=false
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
```

Сервис HawtIO https://srvjx01.somedomain.ru/hawtio позволяет отслеживать параметры java машины в реальном времени. Сервис подключается к JMX порту процесса java приложения.



Мониторинг загруженности ресурсов хоста через web

Мониторинг через web доступен по ссылке имени хоста и пути status, например: https://srvjx01.somedomain.ru/status. Ниже показан примеры экранов:




644
643.8
643.6
15:32:30 15:33:00 15:33:30 15:34:00 15:34:30 15:35:00 15:35:30 15:36:00 15:36:30 15:37:00 15:37:30 15:38:00 15:38:30 15:39:00 15:39:30 15:40:00
KiB
Thu, Dec 15, 2022 | 15:40:29
allocated 644
CPUs
Detailed information for each CPU of the system. A summary of the system for all CPUs can be found at the System Overview section.
utilization
Core utilization (cpu.cpu0)
100.0
80.0
60.0
40.0
20.0
0.0
15:32:30 15:33:00 15:33:30 15:34:00 15:34:30 15:35:00 15:35:30 15:36:00 15:36:30 15:37:00 15:37:30 15:38:00 15:38:30 15:39:00 15:39:30 15:40:00
percentage
Thu, Dec 15, 2022 | 15:40:29
nice 0.0
system 0.8
user 0.0
interrupt 0.0
System Overview
cpu
load
disk
ram
swap
network
processes
idlejitter
interrupts
softnet
uptime
clock synchronization
ipc message queues
ipc semaphores
ipc shared memory
CPUs
Memory
Disks
IPv4 Networking
Network Interfaces
Applications
User Groups
Users
Anomaly Detection


Memory
Detailed information about the memory management of the system.
system
The amount of memory that can be used by user-space processes without causing swapping. Calculated as the sum of free, cached, and inactive memory.
Available RAM for applications (mem.available)
16.9608
16.9590
16.9571
16.9551
16.9531
16.9512
16.9492
15:33:00 15:33:30 15:34:00 15:34:30 15:35:00 15:35:30 15:36:00 15:36:30 15:37:00 15:37:30 15:38:00 15:38:30 15:39:00 15:39:30 15:40:00 15:40:30
GiB
Thu, Dec 15, 2022 | 15:40:57
avail 16.9582
Total amount of real (physical) memory used.
Total Real Memory In Use (mem.real)
System Overview
CPUs
Memory
system
Disks
IPv4 Networking
Network Interfaces
Applications
User Groups
Users
Anomaly Detection
Netdata Monitoring
Add more charts
Add more alarms
Every second, Netdata collects 796 metrics on lnx-jx-app1-d.virtusystems.ru, presents them in 264 charts