Скрипты для взаимодействия с wiSLA

Поддерживаемые параметры установщика Wisla

Чтобы узнать доступные параметры установщика Wisla, выполнить команду:

```
./wisla-5.2.9-2502251017.run --help
```

Будет следующий вывод:

```
wiSLA 5.2.9 build 2502251017 installer
usage: ./wisla-5.2.9-2502251017.run <options>
This script installs the wiSLA system cluster.
OPTIONS:
   Installer options
   -h      Show this message
   -t      Text mode only
   --hadoop-install                  Install hadoop system only
   --silent-install                  Silent install
   --silent-update                   Silent update
   --silent-fast-update              Silent update without backup stage
   --silent-fast-update-with-new-db <Postrgres DB dump path>  Silent update without backup stage with new database restoring
   --silent-fast-update-replace-db-n-hbase <Postgres DB dump path> <HBase data dump path>  Silent update without backup stage with
complete data overwrite

   Package options
   -x      Unpack distribution from self
   -v      Display version information

ENVIRONMENT:
   INSTALL_TEMP - directory used to store temporary files. Default: /tmp/
   INSTALL_LOG  - installation log name. Default: install.YYYY-MM-dd_HH:mm:ss.log
   EXTERNAL_DISTRIBUTION_FILE - used to specify external distribution file
   SCENARIO - used to switch default scenario
```

Варианты обновления системы

1. Установка wisla с нуля

Убедиться что в системе отсутствуют установленные ранее папки : hadoop, postgresql, zookeeper, hbase. Иначе будет полуавтоматический режим установки, где необходимо

будет подтверждать очистку каталогов, а при попытке нажать No процесс установки будет прерван.

```
./wisla-5.2.9-xxxxxxxx.run --silent-install
```

2. Стандартное обновление (с резервным копированием)

```
./wisla-5.2.9-xxxxxxxx.run --silent-update
```

Этот режим выполняет обновление системы с сохранением резервных копий данных.

3. Быстрое обновление без резервного копирования

```
./wisla-5.2.9-xxxxxxxx.run --silent-fast-update
```

Этот режим пропускает этап создания резервных копий, ускоряя процесс обновления.

4. Быстрое обновление с восстановлением новой базы данных PostgreSQL

```
./wisla-5.2.9-xxxxxxxx.run --silent-fast-update-with-new-db /path/to/postgres_dump.sql
```

Здесь `/path/to/postgres_dump.sql` — путь к файлу дампа PostgreSQL, который будет использован для восстановления данных.

5. Полная замена базы данных и HBase

```
./wisla-5.2.9-xxxxxxxx.run --silent-fast-update-replace-db-n-hbase /path/to/postgres_dump.sql /path/to/hbase_data
```

Где:

- `/path/to/postgres_dump.sql` — путь к дампу PostgreSQL.
- `/path/to/hbase_data` — путь к данным HBase.

6. Установка только Hadoop

```
./wisla-5.2.9-xxxxxxxx.run --hadoop-install
```

Дополнительно:

- **Просмотр логов**

По умолчанию логи установки сохраняются в файл `install.YYYY-MM-dd_HH:mm:ss.log`.
Изменить имя лога можно так:

```
export INSTALL_LOG=my_install_log.txt
```

- **Изменение временной директории**

```
export INSTALL_TEMP=/custom/temp/dir
```

- **Режим текстового интерфейса**

```
./wisla-5.2.9-xxxxxxxx.run -t
```

Управление всеми сервисами Wisla

- **Проверка статуса запущенных сервисов**

```
/opt/wisla5/scripts/wisla5.sh status
```

- **Остановка всех сервисов**

```
/opt/wisla5/scripts/wisla5.sh stop
```

- **Запуск всех сервисов**

```
/opt/wisla5/scripts/wisla5.sh start
```

- **Перезапуск всех сервисов**

```
/opt/wisla5/scripts/wisla5.sh restart
```

Управление только сервисом Wisla

- **Проверка статуса запущенных сервисов**

```
/opt/wisla5/scripts/wisla.sh status
```

- **Остановка всех сервисов**

```
/opt/wisla5/scripts/wisla.sh stop
```

- **Запуск всех сервисов**

```
/opt/wisla5/scripts/wisla.sh start
```

- **Перезапуск сервиса Wisla**

```
/opt/wisla5/scripts/wisla.sh restart
```

Скрипт:

```
#!/bin/bash

# Source function library.
WILDFLY_WORK=/opt/wisla5/wildfly/current
wildfly_pid_calc=$(pgrep -u wisla -f "jboss.home.dir=${WILDFLY_WORK}" | wc -l)

keyphrase_blank_wildfly_started="WildFly .* started in"

function start_blank_wildfly() {
        # P°PSP°P»PëP· log-C„P°P№P»P°
        [ -r $WILDFLY_WORK/standalone/log/server.log ] && basic_blank_wildfly_started_counter=$(egrep -c
"keyphrase_blank_wildfly_started" $WILDFLY_WORK/standalone/log/server.log)
```

```
        [ -z "basic_blank_wildfly_started_counter" ] && basic_blank_wildfly_started_counter=0
        # P·P°PïCѓCЃPє PïCѓCЃC,PsPiPs Wildfly
        cd $WILDFLY_WORK/bin
        nohup ./standalone.sh > /dev/null 2>&1&
}

function start_deploy() {
        cd $WILDFLY_WORK/standalone/deployments
        for file in *.?ar;
        do
                touch "$file".dodeploy
        done
}

function wait_for_blank_wildfly() {
    timeout=300
    blank_wildfly_counter_started=0
    blank_wildfly_current_counter_started=0
    if [ ! -e "$WILDFLY_WORK/standalone/log/server.log" ] ; then
        touch "$WILDFLY_WORK/standalone/log/server.log"
    fi
    basic_blank_wildfly_started_counter=$(egrep -c "$keyphrase_blank_wildfly_started" $WILDFLY_WORK/standalone/log/server.log)
    for i in $(seq 1 $timeout);
    do
        blank_wildfly_started_counter=$(egrep -c "$keyphrase_blank_wildfly_started" $WILDFLY_WORK/standalone/log/server.log)
        ((total_wildfly_start_time = total_wildfly_start_time + 1))
        if [ $blank_wildfly_started_counter -eq $((basic_blank_wildfly_started_counter+1)) ]
        then
            echo "Starting blank Wildfly..."
            return
        fi
    sleep 1
    done
    echo "Error during blank Wildfly start!"
}

function start() {
        wildfly_pid_calc=$(pgrep -u wisla -f "jboss.home.dir=${WILDFLY_WORK}" | wc -l)
        if [ "$wildfly_pid_calc" -eq 0 ]
        then
                total_wildfly_start_time=0
                # PsC‡PëCЃC,PєP° PєP°C,P°P»PsPiPsPI deployments (PєCЂPsPjPµ war, ear) Pë tmp PïPµCЂPµPr P·P°PïCѓCЃPєPsPj
                find $WILDFLY_WORK/standalone/deployments -regextype posix-egrep ! -regex ".*(ear|war)" -type f -exec rm -f {} \;
                rm -rf $WILDFLY_WORK/standalone/tmp/*
                rm -rf /tmp/workspace_*-*-*-*-*
                cd $WILDFLY_WORK/standalone
                # P·P°PïCѓCЃPє PïCѓCЃC,PsPiPs wildfly
                echo "Waiting for the blank Wildfly application server to start (up to 5 minutes)... "
                start_blank_wildfly
                # PsP¶PëPrP°PSPëPµ P·P°PïCѓCЃPєP° PïCѓCЃC,PsPiPs Wildfly
                wait_for_blank_wildfly
                # PSP°C‡P°P»Ps PrPµPïP»PsCЏ P°CЂC,PµC„P°PєC,PsPI (PI PïPsCЂCЏPrPєPµ, PIC‹P±CЂP°PSPSPsPj Wildfly)
                start_deploy
        else
```

```
        echo "Error. Wildfly is already running!"
                return 1
        fi
}


function stop() {
    echo "wiSLA is stopping..."
    kill `pgrep -f "jboss.home.dir=${WILDFLY_WORK}"` &> /dev/null
    ATTEMPTS=0
    while [[ ! -z `pgrep -f "jboss.home.dir=${WILDFLY_WORK}"` ]]; do
        echo "Waiting for the Wildfly application server to stop..."
        (( ATTEMPTS = ATTEMPTS + 1 ))
        if [ $ATTEMPTS -gt 10 ]
            then
            echo "60 seconds is elapsed, trying to stop the process with -9 signal"
            kill -9 `pgrep -f "jboss.home.dir=${WILDFLY_WORK}"`
        fi
        sleep 5
    done
}

function status() {
    PID=`pgrep -f "jboss.home.dir=${WILDFLY_WORK}"`
    [[ ! -z $PID ]] && echo "Started with PID : $PID" || echo "Stopped"
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        ;;
    try-restart)
        stop
        start
        ;;
    status)
        status
        ;;
    *) exit 1
esac

exit 0
```

Реиндексация

- **Стандартная реиндексация (требует запущенной системы)**

```
/opt/wisla5/scripts/wi-reindex.sh
```

Скрипт:

```
#!/bin/bash
# Source function library.
JRE_WORK="/opt/wisla5/jre/current"

echo "Reindexing wisla engine lucene database..."
${JRE_WORK}/bin/java -jar /opt/wisla5/util/jmx/cmdline-jmxclient-0.10.3.jar - localhost:1090 "wisla-engine:name=nodeReindexer"
reindexFullTextSearch

exit 0
```

- **Независимая реиндексация (не зависит от состояния деплоя)**

```
/opt/wisla5/scripts/wi-reindex-standalone.sh
```

Скрипт:

```
#!/bin/bash
# Source function library.
JRE_WORK="/opt/wisla5/jre/current"
POSTGRES_HOST="alfa-test"
DB_NAME="wisla"
USER="wisla"

echo "Reindexing database"
${JRE_WORK}/bin/java -jar -Dhibernate.connection.username="${USER}" \
-Dhibernate.search.base_indexes_directory="/opt/wisla5/wildfly/current/bin/searchindexes/engine/" \
-Dhibernate.connection.url="jdbc:postgresql://${POSTGRES_HOST}:5462/${DB_NAME}" \
-Dcom.sun.xml.bind.v2.bytecode.ClassTailor.noOptimize=true \
/opt/wisla5/util/reindexer/wisla-reindexer.jar
echo "Reindex finished"
exit 0
```

Создание резервных копий баз данных

- **PostgreSQL**

Файл шаблона скрипта: `/opt/wisla5/backup_scripts/postgres_backup_template.sh`

Пример запуска резервного копирования:

```
pg_dump --host "wisla" --port 5432 --username "wisla" --format custom --blobs --no-owner --encoding UTF8 --verbose --file /home/wisla/backup/wisla.backup
```

Альтернативный способ через шаблонный скрипт:

```
sed "s|{{FILE-NAME}}|/home/wisla/backup/wisla.backup|" /opt/wisla5/backup_scripts/postgres_backup_template.sh | bash
```

Скрипт:

```
#!/bin/bash
pg_dump --host "alfa-test" --port 5432 --username "wisla" --format custom --blobs --no-owner --encoding UTF8 --verbose --file {{FILE-NAME}}
wisla
ssh {{LOGIN}}@{{BACKUP-SERVER}} "mkdir -p {{DESTINATION}}"
scp ./{{FILE-NAME}} {{LOGIN}}@{{BACKUP-SERVER}}:{{DESTINATION}}
rm -f ./{{FILE-NAME}}
```

- **HBase**

Файл шаблона скрипта: `/opt/wisla5/backup_scripts/hbase_backup_template.sh`, но корректность работы проверить не удалось, пробовал переписать скрипт, но возникает проблема при выполнении команды импорта снимков в папку.

```
#!/bin/bash

BACKUP_DIR=$(mktemp -d -t -p ~/ hbase_backup.XXXXXXXX)
mkdir ${BACKUP_DIR}/backup
BACKUP_PARRENT_DIR=${BACKUP_DIR}
BACKUP_DIR=${BACKUP_DIR}/backup

hbase org.apache.hadoop.hbase.mapreduce.Export -D mapred.output.compress=true {{TABLE-PREFIX}}-tsdb {{TABLE-PREFIX}}-tsdb-backup 1 {{START-TIMESTAMP}} {{END-TIMESTAMP}}
hbase org.apache.hadoop.hbase.mapreduce.Export -D mapred.output.compress=true {{TABLE-PREFIX}}-tsdb-uid {{TABLE-PREFIX}}-tsdb-uid-backup 1 {{START-TIMESTAMP}} {{END-TIMESTAMP}}
hbase org.apache.hadoop.hbase.mapreduce.Export -D mapred.output.compress=true {{TABLE-PREFIX}}-tsdb-nf {{TABLE-PREFIX}}-tsdb-nf-backup 1 {{START-TIMESTAMP}} {{END-TIMESTAMP}}
hbase org.apache.hadoop.hbase.mapreduce.Export -D mapred.output.compress=true {{TABLE-PREFIX}}-tsdb-uid-nf {{TABLE-PREFIX}}-tsdb-uid-nf-backup 1 {{START-TIMESTAMP}} {{END-TIMESTAMP}}
hbase org.apache.hadoop.hbase.mapreduce.Export -D mapred.output.compress=true {{TABLE-PREFIX}}-tsdb-lts {{TABLE-PREFIX}}-tsdb-lts-backup 1 {{START-TIMESTAMP}} {{END-TIMESTAMP}}

hdfs dfs -copyToLocal /user/wisla/{{TABLE-PREFIX}}-tsdb-backup ${BACKUP_DIR}/{{TABLE-PREFIX}}-tsdb
hdfs dfs -copyToLocal /user/wisla/{{TABLE-PREFIX}}-tsdb-uid-backup ${BACKUP_DIR}/{{TABLE-PREFIX}}-tsdb-uid
hdfs dfs -copyToLocal /user/wisla/{{TABLE-PREFIX}}-tsdb-nf-backup ${BACKUP_DIR}/{{TABLE-PREFIX}}-tsdb-nf
hdfs dfs -copyToLocal /user/wisla/{{TABLE-PREFIX}}-tsdb-uid-nf-backup ${BACKUP_DIR}/{{TABLE-PREFIX}}-tsdb-uid-nf
hdfs dfs -copyToLocal /user/wisla/{{TABLE-PREFIX}}-tsdb-lts-backup ${BACKUP_DIR}/{{TABLE-PREFIX}}-tsdb-lts

cd ${BACKUP_DIR}/..
tar -czvf {{FILE-NAME}} backup
hdfs dfs -rm -r /user/wisla/{{TABLE-PREFIX}}-tsdb-backup
hdfs dfs -rm -r /user/wisla/{{TABLE-PREFIX}}-tsdb-uid-backup
hdfs dfs -rm -r /user/wisla/{{TABLE-PREFIX}}-tsdb-nf-backup
hdfs dfs -rm -r /user/wisla/{{TABLE-PREFIX}}-tsdb-uid-nf-backup
```

```
hdfs dfs -rm -r /user/wisla/{{TABLE-PREFIX}}-tsdb-lts-backup

ssh {{LOGIN}}@{{BACKUP-SERVER}} "mkdir -p {{DESTINATION}}"
scp ./{{FILE-NAME}} {{LOGIN}}@{{BACKUP-SERVER}}:{{DESTINATION}}

rm -f ./{{FILE-NAME}}
cd ../
rm -r ${BACKUP_PARRENT_DIR}
```

!!!Но для выполнения скрипта надо настраивать hbase и hadoop

Дополнительно:

- **Перезагрузка плагинов в приложении wiSLA**

```
/opt/wisla5/scripts/reload-plugins.sh
```

Скрипт:

```
#!/bin/bash
# Source function library.
JRE_WORK=/opt/wisla5/jre/current

${JRE_WORK}/bin/java -jar /opt/wisla5/util/jmx/cmdline-jmxclient-0.10.3.jar - localhost:1090 "wisla-engine:name=pluginManager"
reloadPlugins

exit 0
```

- **Удаление старых лог файлов**

```
/opt/wisla5/scripts/remove-old-logs.sh
```

Скрипт:

```
#!/bin/bash
DAYS=10
LOG_DIR=/opt/wisla5/wildfly/current/standalone/log
for file in "$( find ${LOG_DIR}/ -maxdepth 1 -type f -mtime +${DAYS} )"
do
    rm -f ${file}
done
```