Cronjobs einrichten

Viele Aufgaben können in i-doit durch das CLI Tool Controller automatisiert werden. Dies machen wir uns zu Nutze, um die IT-Dokumentation regelmäßig zu warten.

> **Einrichten ja oder nein?**
>
> Die Einrichtung von Cronjobs ist optional, doch es gibt eine klare Empfehlung, diese auf jeden Fall einzurichten – bestenfalls direkt nach der Installation.

Inhaltsverzeichnis

Aufruf vom Controller vereinfachen

Um den Controller simpel aufzurufen, eignet sich ein einfaches Bash-Skript:

```
sudo nano /usr/local/bin/idoit
```

Das Script bekommt folgenden Inhalt, der an die eigene Installation angepasst werden muss:

```
#!/bin/bash

##
## i-doit console
##

##
## Copyright (C) 2017-18 synetics GmbH, <https://i-doit.com/>
##
## This program is free software: you can redistribute it and/or modify
## it under the terms of the GNU Affero General Public License as published by
## the Free Software Foundation, either version 3 of the License, or
## (at your option) any later version.
##
## This program is distributed in the hope that it will be useful,
## but WITHOUT ANY WARRANTY; without even the implied warranty of
## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
## GNU Affero General Public License for more details.
##
## You should have received a copy of the GNU Affero General Public License
## along with this program. If not, see <http://www.gnu.org/licenses/>.
##

set -euo pipefail

##
## Configuration
##

INSTANCE_PATH="/var/www/html/i-doit"
APACHE_USER="www-data"
ARGS="$*"

##--------------------------------------------------------------------------------------------------

function execute {
    local prefix=""
    local console="php console.php $ARGS"

    test "$(whoami)" != "$APACHE_USER" && prefix="sudo -u $APACHE_USER "

    eval "${prefix}${console}" || abort "i-doit console exited with non-zero status"
}

function setup {
    cd "$INSTANCE_PATH" || abort "No i-doit instance found under '${INSTANCE_PATH}'"
}

function finish {
    exit 0
}

function abort {
    echo -e "$1"  1>&2
    echo "Operation failed. Please check what is wrong and try again." 1>&2
    exit 1
}

function log {
    echo -e "$1"
}

##--------------------------------------------------------------------------------------------------

if [[ "${BASH_SOURCE[0]}" = "$0" ]]; then
    setup && execute && finish
fi
```

Das Script wird anschließend ausführbar gemacht:

```
sudo chmod +x /usr/local/bin/idoit
```


sudo

Das Script erlangt automatisch die richtigen Rechte durch den Einsatz von **sudo**.

Dem derzeit angemeldeten User muss daher das Ausführen von **sudo** erlaubt sein. Unter Debian GNU/Linux 9 wird der User der Gruppe **sudo** zugewiesen. Dazu sind **root**-Rechte erforderlich:

```
usermod -aG sudo "$(whoami)"
```

Unter SLES ist die Option **Defaults targetpw** gesetzt, die diesen Mechanismus verhindert. Daher muss diese Option in der Datei **/etc/sudoers** auskommentiert werden.

Ab dann kann es von jedem User verwendet werden:

```
idoit COMMAND [OPTIONEN]
```

Essentielle Jobs anlegen

Im nächsten Schritt legen wir ein weiteres Script an, das wir sowohl manuell als auch automatisiert aufrufen können:

```
sudo nano /usr/local/bin/idoit-jobs
```

Dieses Script bekommt folgenden Inhalt:

```
#!/bin/bash

##
## i-doit jobs
##

##
## Copyright (C) 2017-18 synetics GmbH, <https://i-doit.com/>
##
## This program is free software: you can redistribute it and/or modify
## it under the terms of the GNU Affero General Public License as published by
## the Free Software Foundation, either version 3 of the License, or
## (at your option) any later version.
##
## This program is distributed in the hope that it will be useful,
## but WITHOUT ANY WARRANTY; without even the implied warranty of
## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
## GNU Affero General Public License for more details.
##
## You should have received a copy of the GNU Affero General Public License
## along with this program. If not, see <http://www.gnu.org/licenses/>.
##

set -euo pipefail
IFS=$'\n\t'

##
## Configuration
##

CONSOLE_BIN="/usr/local/bin/idoit"
INSTANCE_PATH="/var/www/html/i-doit"
APACHE_USER="www-data"
IDOIT_USERNAME="admin"
IDOIT_PASSWORD="admin"
TENANT_ID="1"
```

```
##----------------------------------------------------------------------------------------------------

function execute {
    local prefix=""
    local suffix="--user $IDOIT_USERNAME --password $IDOIT_PASSWORD --tenantId $TENANT_ID"

    test "$(whoami)" != "$APACHE_USER" && prefix="sudo -u $APACHE_USER "

    log "Archive i-doit logbook"
    eval "${prefix}${CONSOLE_BIN} logbook-archive $suffix" || \
        abort "Command 'logbook-archive' failed"
    log ""

    log "Cleanup i-doit rights"
    eval "${prefix}${CONSOLE_BIN} auth-cleanup $suffix" || \
        abort "Command 'auth-cleanup' failed"
    log ""

    log "Purge unfinished objects"
    eval "${prefix}${CONSOLE_BIN} system-objectcleanup --objectStatus 1 $suffix" || \
        abort "Command 'system-objectcleanup' failed"
    log ""

    log "Re-create search index"
    eval "${prefix}${CONSOLE_BIN} search-index $suffix" || \
        abort "Command 'search-index' failed"

    log "Send notifications"
    eval "${prefix}${CONSOLE_BIN} notifications-send $suffix" || \
        abort "Command 'notifications-send' failed"

    log "Clear caches"
    eval "${prefix}rm -rf ${INSTANCE_PATH}/temp/*" || \
        abort "Unable to clear caches"

    log "Clear updates"
    eval "${prefix}rm -rf ${INSTANCE_PATH}/updates/versions/*" || \
        abort "Unable to clear updates"
}

function setup {
    test -x "$CONSOLE_BIN" || \
        abort "Script '${CONSOLE_BIN}' not found"

    test -d "$INSTANCE_PATH" || \
        abort "No i-doit instance found under '${INSTANCE_PATH}'"
}

function log {
    echo -e "$1"
}

function finish {
    log "Done. Have fun :-)"
    exit 0
}

function abort {
    echo -e "$1"  1>&2
    echo "Operation failed. Please check what is wrong and try again." 1>&2
    exit 1
}

##----------------------------------------------------------------------------------------------------

if [[ "${BASH_SOURCE[0]}" = "$0" ]]; then
    setup && execute && finish
fi
```

Das Script wird anschließend ausführbar gemacht:

```
sudo chmod +x /usr/local/bin/idoit-jobs
```

Ab dann kann es von jedem User verwendet werden:

```
idoit-jobs
```

Wird das Script ausgeführt werden folgende Arbeiten erledigt:

- Datei-Caches im **`temp/`**-Verzeichnis werden geleert.
- Nicht mehr benötigte Update-Pakete werden gelöscht.
- Ältere Logbuch-Einträge werden archiviert.
- Der Cache für Benutzerrechte wird neu aufgebaut.
- "Unfertige" Objekte werden unwiderruflich gelöscht.
- Der Suchindex wird neu aufgebaut.
- Benachrichtigungen werden per E-Mail versendet.

Aufruf der Jobs automatisieren

Wann und wie oft?

Es empfiehlt sich, die oben genannten Jobs mindestens einmal pro Tag auszuführen. Es sollte sichergestellt sein, dass während der Ausführung keine weiteren Interaktionen in i-doit geschehen, sprich weder über die Web GUI, durch zusätzliche Scripte oder von externen Applikationen über die API. Daher werden die Jobs meist nachts ausgeführt.

GNU/Linux

Unter Linux können Befehle automatisiert in regelmäßigen Abständen ausgeführt werden. Hierzu bieten sich cron, anacron, crontab oder systemd.timer an.

Windows

Die in diesem Artikel genannten Scripte funktionieren nicht ohne weiteres unter Windows und sollten durch äquivalente Scripte ersetzt werden. Für die Automatisierung eignen sich Windows Tasks.

Probleme abfangen

Damit ggf. Fehler abgefangen und an den zuständigen Systemadministrator gemeldet werden, kann man das Betriebssystem so konfigurieren, dass E-Mails verschickt werden. Ein simpler Mailer ist **`sSMTP`**. Das Script **`idoit-jobs`** produziert allerdings eine Menge Ausgaben auf der Shell. Damit nur Fehler gemeldet werden, kann man das Tool **`chronic`** einsetzen, das sich unter vielen Betriebssystemen nachinstallieren lässt. Meist ist es im Distributionspaket **`moreutils`** enthalten.

chronic unter SLES

Das Paket **`moreutils`** und somit das Tool **`chronic`** sind kein Bestandteil von SLES. Daher muss **`chronic`** manuell von der Website heruntergeladen werden:

```
wget https://git.joeyh.name/index.cgi/moreutils.git/plain/chronic
chmod +x chronic
sudo mv chronic /usr/bin/
wget -O - https://cpanmin.us | perl - --sudo App::cpanminus
sudo cpanm --notest --install IPC::Run
```

Beispiel für Cron

Für Cron legen wir eine neue Datei an:

```
sudo nano /etc/cron.d/i-doit
```

Diese Datei befüllen wir mit folgendem Inhalt:

```
## i-doit cron jobs

MAILTO="sysadmin@i-doit.example.net"

5 5 * * *   www-data    test -x /usr/local/bin/idoit-jobs && /usr/bin/chronic /usr/local/bin/idoit-jobs
```

Jeden Tag um 5:05 Uhr wird mit den Rechten des Apache Users **`www-data`** das Script **`idoit-jobs`** ausgeführt. Damit nur Fehler per Mail gesendet werden, kommt **`chronic`** zum Einsatz. Fehler werden per Mail versendet.