**Customized SSH-tool – launcher.sh**

Index

Contents

[1 Introduction 3](#_Toc17881258)

[2 General Processing 3](#_Toc17881259)

[3 Connection 4](#_Toc17881260)

[4 SUDO 4](#_Toc17881261)

[5 Target Systems 5](#_Toc17881262)

[6 Logging 5](#_Toc17881263)

[7 Local Script 5](#_Toc17881264)

[8 Input variables: 6](#_Toc17881265)

[9 Wish List 8](#_Toc17881266)

# Introduction

This document provides some detailed insights regarding the “Customized SSH-tool – launcher.sh” (a.k.a. *anthoble*).

The tool is currently operational in order to execute commands/scripts to 1 or more (up to unlimited) systems parallel; at the same time, initialized from 1 centralized point/system.

In this case:

* kl121hal

The tool is located on:

* kl121hal@/tech/local/bin/launcher.sh

Operational version:

* 3.0.0 - Aug 28th 2019

Written by:

* Anthony Paardekooper

# General Processing

The tool is set up in such a way that it executes things, scripts or commands, on a maximum of 150 systems in parallel (in the background). The number of 150 is not hard limited, but more chosen to make it "*not too much*"; the tool uses file descriptors, so basically as many operations could be performed at the same time as that file descriptors are available per user.  
However, that will probably have a noticeable impact on the system (kl121hal) where the tool was started on. Hence the default for the tool is to handle 30 systems in parallel; however, it is possible to override the default of 30 by using/specifying an input parameter.

That input parameter (-n) must then contain a value between 1 and 150.

The approach (since version 3.0.0) is to have continuously the maximum number of allowed background processes to be running: as soon as a file-descriptor becomes available, define the status of the process/system such as was running in it and start a new 1 in the file-descriptor. This is different behavior as the formerly used “batchwise processing”; then the total number of systems was split into batches and the batch should be completed before any new batches could be started.

The tool itself monitors the operations that have been started to/on the specified systems: if these are still operational after 10 minutes (=default, this threshold can be overridden by an input parameter (–m), any processes still running after time will be terminated and marked as failed.  
Therefore it is not possible that the tool runs into “a hanging state”.  
  
The tool determines per system on which a command/script has been started and what the status thereof is: it is not the case that some operations are “just thrown away” and at the result is looked at.  
When all specified systems have been handled, a clear overview is given at the end, regarding:

* Systems were unavailable.
* Systems on which the operations have been successfully performed.
* Systems on which the operations did not produce a good result.

# Connection

The connection to the systems on which the operations must be performed is based on SSH (after it was established that the system is available by using the tcping-command).  
It is not conditional that the SSH keys are first distributed, but if that is the case, the connection will of course work. The tool requires a password with which the SSH connection is set up; this is not shown in clear text and is also not visible in the process list on the system on which the tool was started.  
The tool works in such a way that the first system from the list of systems of operations is used to test whether a successful SSH connection can be set up. If this is not the case, the tool does not continue; after all, there is a risk that an incorrect password has been specified and if several systems are triggered with an incorrect password, the user with whom the tool was started is blocked.

# SUDO

The tool uses SUDO by default (=sudo -ni and then performs the operations on the target system(s)), for which it is of course important that the correct password is specified!

Here too, the password is not visible in clear text and is first tested on the first system before proceeding.  
  
If no\_password is provided as a password, this will be recognized by the tool and no issues regarding authentication via passwords will be carried out; for this it is important that the SSH keys of the relevant user are distributed and that corresponding SUDO rules are present for which no password is required.  
This is especially important for scheduling (=no interactive sessions), then there is no need to save a valid password (other than: -p no\_password).

The following SUDO rules have now been set for the user sftuser with which the SUDO distribution can be performed:

User\_Alias ​​LIN\_SUDODISTRIBUTION\_USER = sftuser  
Host\_Alias ​​LIN\_SUDODISTRIBUTION\_HOSTS = ALL  
LIN\_SUDODISTRIBUTION\_USER LIN\_SUDODISTRIBUTION\_HOSTS = (root) NOPASSWD: \ /usr/bin/id -un  
LIN\_SUDODISTRIBUTION\_USER LIN\_SUDODISTRIBUTION\_HOSTS = (root) NOPASSWD: \ /tech/local/sbin/wget-sudoers.sh

It is also possible that the tool does not use SUDO; all operations are then started on the "target system(s)" with the user with whom they were started.

That is how the tool very often is used (and like the *new SUDO merger interpretation for determining the WebSystems Host\_Aliases* is currently implemented) because of having no/limited SUDO access.

In the case of using SUDO, a short *answering game* is played first by the tool itself the on target-system(s) to allow the validation/authorization to take place (for executing: sudo -ni); only then will the actual command/script be started as root.

The tool is setup in such a way, that either the -D (do-not-use-sudo) or -S (do-use-sudo) input-parameters must be passed on.

When using, more than 1 command/script (“<cmd>;<cmd>”) to be executed on the target system(s), only the 1st command will be excuted as root when the –S input-parameter was passed on. If it is required that all commands should be executed as root, the following syntax should be used:

* “<cmd>;sudo <cmd>;sudo <cmd>”

**Please do keep in mind**: When using “multiple remote commands”, as shown above, only the status of the last executed command/script will be captured. And that 1 actually defines the status of the “entire run” for a specific system.

# Target Systems

The systems can be retrieved from the CMDB (LMIS) on the basis of selections on which the operations must be performed. Or just one or more systems can be specified on the tool, on which the operations must be performed.

# Logging

The tool keeps track of logging for every time it is started, and displays what operations are being performed, what the status is and the "final report" regarding the "grand totals". This logging takes place per user and per session so that log files are not overwritten and no conflicts arise due to permission problems (in the case that multiple users use the tool).  
What is shown on the screen (can contain a lot of info when the tool performs operations for many systems) and as such it is also stored in the log file.

In case of the -L option was specified (=<use-local-copy>), a copy of the local script will also be displayed in the log-file.

# Local Script

The tool also has the ability to start a local script (from the kl121hal) on all target systems; in that case the local script will be copied to the target system via SCP, and then be started on the target system. The script then is placed on the target system in:

* /tmp/.<id -un>/$$/<script name>

This is to prevent permission problems arising and that existing scripts on the target system might be overwritten.

# Input variables:

The following input variables are recognized by the tool (some are mandatory and some are optional) and may provide some more clarity about how the tool works:

* -x "<command(s)/script(s)> "

Operation(s) as it should be started on the target system(s).

From the options below, 1 of the 5 options must be specified:

* -s "<system(s)>"  
  List of system (s) on which operations must be started; can of course also be the  
  output of a command are:
* "$(cat /tmp/.system\_list)"
* -e <AE|AVE|DE|EE|LE|UTE|SE|TE|ITE>  
  The environment on which the operations must be performed.
* -f <file>  
  File containing the system(s) on which operations must be started.
* -t PLANNED - Operations are started on all **PLANNED** systems  
  -t OPERATION - Operations are started on all **IN OPERATION** systems
* –A - Operations are started on **ALL** systems

From the options below, either 1 option must be specified:

* -D  
  Do not use SUDO; this flag does not perform the operation as root.
* -S  
  Do use SUDO; SUDO (sudo -ni) is used by default, this flag does indeed perform the operation as root (with SUDO).
* -p <password>  
  The password with which the SSH connection is set up, and when the operations must be performed as root, is used for the SUDO validation as well.  
  If the SSH keys are distributed for the user with which the tools was started, this can contain a *bogus password* because the SSH keys are then used. That of course does not apply to the SUDO validation!   
  Although when this input parameter is used, this parameter is changed by the tool in the command history (to: zzzzzzzz), specifying a password from the command-line remains tricky ...

If the password contains the value (literally!): no\_password, no authentication via passwords is done and it is assumed that for any sudo actions, **SUDO rules exist without passwords** on the target system on which they are executed.

* -L  
  The script to be started on the target systems exists locally on the kl121hal and will therefore first be copied to the target system before being started.
* -N  
  Do not ask to confirm that the specified list of systems is correct; perform the operations on it immediately (important for when scheduled operations are performed).
* -Q  
  Quiet mode; show virtually nothing on the screen, only the log file used for the session.
* -n <max processes>  
  The number of processes (systems) that are started per batch in parallel to each other; just as many batches until all systems have been processed.  
  The value must contain a number between 1 and 150.
* -m <seconds>  
  The maximum number of seconds that the operation (per target system) may take.  
  If that is exceeded, the process will be terminated.
* -w <waiting seconds>  
  The number of seconds (default=5) to be waited before checking process(es) are still running. It is not allowed the sleep-time to be greater than the number of seconds an individual processes may take; in that case it will be automatically set to: ${MAX\_SECS}

If the required/correct input variables are not given to the tool, this will be shown in the following way:

\*\*ERROR\*\* usage: launcher.sh

-x "<command to execute>"

-A (=ALL; Planned and Operation) |  
 -t <PLANNED | OPERATION> |

-e <AE | AVE | DE | EE | LE | UTE | SE |TE | ITE> |

-s "<system> <system> <system> ..." |

-f <file>

-D (=DO not use sudo ) | -S (=do use SUDO)

[ -m <max seconds> ]  
 [ -n <max processes> ]

[ -p <password> ]

[ -w <waiting seconds>]

[ -L ] (=use LOCAL copy )

[ -N ] (=NO confirmations asked )

[ -Q ] (=QUIET mode )

# Wish List

Despite the tool has an operational state, there are still some things which might be improved.

Therefore the following items are noted on the “wish list”:

* **Logging**:

Making it impossible for users to edit their own logging of the tool.  
(Write the logging elsewhere: splunk?)

Possibility:   
touch -t $(date +%s | awk '{ date=$1-90 ; print strftime("%Y%m%d%H%M",date) }') /tmp/.flag\_file  
find /tmp/.rem-exec/ -type f ! -newer /tmp/.flag\_file -name ".launcher.sh.[0-9]\*" -size +0 | xargs -I {} lsattr {} | awk 'substr($1,5,1) !~ /^i$/ { system("echo chattr +i "$2 ) }'

* **Max-utilization**:

Maximum number of sessions/systems in a batch that are operational at the same time.  
Don't wait until the last system is ready before a new batch is started: if a file descriptor is free, immediately start a new one.  
The technical model must then look different.  
Hint: RUNNING[${NUM}]=$$:$(date +% s):${NUM} and an array with free file descriptors and an array with descriptors whose status is to be determined.

* **Ranges**:

Allow to select multiple systems in the selection screen.

* **-f <systems> :**

Building in the ability the usage of –f input-variable which specifies a file in which the systems are included for which there are operations must be started on.

* **Multiple commands**:

Investigate whether it is possible to execute multiple commands with the sudo feed; technical model will then have to change (sudo –ni vs. sudo - su).  
But very much the question whether that is desired and technically feasible in the  
current setup; and whether that is desirable at all.

* Not required anymore: when it is required that commands, other than the first  
  (i.e.: “id –un; date ; sudo id –un”). Then it works just fine by explicitly stating the command/script proceeded by sudo.
* **Default**:

By default, the tool is set up to perform operations with sudo; perhaps it is better not to use sudo by default, but the other way around. Never use sudo unless...

* **First-time**:

If the user (=no directory in /tmp/.rem-exec) is using the tool for the first time, show a warning about careful handling regarings the passwords (locking etc.) and the usage.

* **~~Encapsulation~~**~~:~~

~~Instead of the current optional encapsulation of command(s)/script(s) and system(s) by   
 double quotes:~~

* + ~~–x "<cmd>" –s "<systems>"~~

~~Have that replaced by the usage of square-brackets and make the usage mandatory:~~

* + ~~-x [cmd] –s [systems]~~
* Seems not to be feasible and as such removed from the wishlist; in case of using square-brackets with multiple commands (semicolon used as the delimiter) just the 1st command will be executed. Therefore it is better to not have the square-brackets solution used. Command(s)/script(s) to be executed on system(s) should be given between double quotes; the usage cannot be enforced,