Securing Operations over SSH in Flux

Flux Announces SSH Command, SSH File Upload, and SSH File Download Actions

Reading Time: 7 minutes

Flux 8.0.10 implements support for SSH based interactions with remote servers in your enterprise workflows. SSH actions can be very handy when it comes to environments where public key infrastructure is enforced and where customers are required to isolate the processes that Flux workflows orchestrate without leveraging the underlying security infrastructure. This will be no longer a case with the introduction of SSH actions in this Flux release. Workflows and file orchestrations now interact cleanly with the enterprise's existing public key infrastructure while still enjoying Flux's powerful integration capabilities.

A Flux workflow can now run commands on remote machines or perform file transfers to or from remote machines. It supports password based authentication as well as public key authentication with passphrase. The following new Flux Actions are available in this release.

- SshCommandAction - runs commands or scripts on remote server
- SshFileUploadAction - uploads file or directories to remote server
- SshFileDownloadAction - downloads file or directories from remote server

SSH interaction provides a secure and very powerful capability in use by many IT teams and production-hardened for decades in infrastructure automation.

Getting Started

Setting up key based authentication between two machines (for example, a desktop Mac Pro and a remote Ubuntu virtual machine) is straightforward. The first step is to generate ssh key pair on the Mac Pro with passphrase:

```
macpro:.ssh arul$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/arul/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/arul/.ssh/id_rsa.
```

```
Your public key has been saved in /Users/arul/.ssh/id_rsa.pub.
The key fingerprint is:
a3:2e:03:c0:0b:98:ed:05:6d:00:a4:60:f6:6f:0c:e3 arul@macpro
The key's randomart image is:
+--[ RSA 2048]----+
|+=.o           |
|= o o          |
|+o *           |
|=.o *          |
|.o.E +  S      |
| .o .  . .     |
|   . .         |
|    o.         |
|     o.        |
+-----------------+
```

Copy the public key (id_rsa.pub) of Mac Pro to the Ubuntu machine.

```
$ scp ~/.ssh/id_rsa.pub ubuntu-vm:
```

Ssh to the Ubuntu machine and append the copied public key to authorized keys file.

```
$ ssh ubuntu-vm
arul@ubuntu-vm's password:
$ mkdir ~/.ssh
$ chmod 700 ~/.ssh
$ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
$ rm ~/id_rsa.pub
$ chmod 600 ~/.ssh/authorized_keys
```

Perform the same setup from the Ubuntu machine to Mac Pro if bi-directional key-based communication over SSH is required. Public key based authentication - encrypted using passphrase - is enabled once the keys are setup on both the machines . You can disable password authentication in your SSH server config, so you avoid sending your clear text password over network.

This setup allows the Mac Pro user to connect to a Linux server and automate tasks without prompting passwords, thus making interaction seamless. This is a secure and very powerful technique used by IT teams and has been battle-tested for decades in infrastructure automation.

SSH actions extend Flux's robust palette of workflow and file actions to increase the power and flexibility of Flux workflows. One can orchestrate a workflow to manage their farm of servers and control them from Flux. Some of the common tasks include, on-demand backups, file transfers, system monitoring, etc. This technique also enables agent-less scheduling. Without requiring any special software on these remote servers, one can run and manage scripts just like any native application using Flux.

Running a Command using an SSH Command Action

Flux actions are configured either using the Flux Java API or via editing dialogs within the Flux web-based user interface. Hostname and username are required properties on these SSH actions. The recommended way to configure the hostname and username is are using Flux's runtime configuration properties – and define these values as reusable variables. This allows reusability of configuration across workflows and ability to make changes at runtime without restarting a Flux engine.

Here is how the runtime.properties file look like in this setup.

```
/host=ubuntu-vm
/username=arul
/fingerprint=e4:af:1f:a3:8b:c0:15:33:71:87:d8:57:8f:d4:1a:3d
/private_key=/Users/arul/.ssh/id_rsa
/passphrase=+KYFTbEq6xaiUwa2Ij4N/Q==
```

The passphrase should be encrypted using Flux's CLI API so no clear text password is stored in the runtime configuration. Here is how to generate the encrypted password using Flux API.

```
$ java -cp flux.jar flux.Main encryptpassword flux*help
Flux 8.0.10 (build #1296) ~ Copyright 2015 Flux Corporation

Encrypted password is +KYFTbEq6xaiUwa2Ij4N/Q==
```

Once the runtime variables are defined, the Flux SSH Action dialog can be populated with the following values.

Property	Value
Hostname	${runtime host}
Username	${runtime username}
Host key Verifier	${runtime fingerprint}

Property	Value
Private Key	${runtime private_key}
Passphrase	${runtime passphrase}
Remote Command	tail -20 /var/log/auth.log

The remote command tails the last 20 lines of the SSH server log. The output of this action is similar to Flux's Process action and contains the following:

- SshCommandActionResult has the result after executing the action.
- result - the process's exit code
- stderr - the standard error from the process.
- stdout - the standard output from the process.

This is analogous to running this SSH command:

```
ssh ubuntu-vm 'tail -20 /var/log/auth.log'
```

Flux's SSH Command Action opens up a range of integration use cases. In addition to performing the remote command, The SSH Command Action also provides a means to manage these remote SSH sessions in case of an error or timeout. Like a Flux Process Action, the SSH Command Action can destroy a session on a signal, timeout or interrupt. This feature which provides a clean way to exit or provide automated error and recovery handing from a workflow involving SSH actions.

File Transfers using SSH File Download and Upload Actions

Flux's new SSH File Download Action allows file transfers over SSH from one server to another server that is configured with public key authentication using SSH. In addition to the required properties for SSH action, one needs to specify the remote path of the file being downloaded and the destination location where to place the file.

Property	Value
Remote Path	/home/arul/Downloads/flux-8-0-10.zip
Local Path	/Users/arul/work/staging

In case of an SSH File Upload Action, one needs to specify the local path of the file being uploaded and

the target location where to place the file.

Property	Value
Local Path	/Users/arul/work/staging/flux-8-0-10.zip
Remote Path	/home/arul/Downloads

The output of these action return the following information:

- SshFileActionResult has the result after executing the action.
- filenames - the list of files that were successfully copied.
- successful_count - the number of files that succeeded file copy.

Both these actions support Zlib based compression for file transfers, which improves transfer speeds.

Remote Process Management using SSH Command Actions

One can now perform remote process management using Flux SSH command actions. Since SSH does not provide a mechanism to manage processes remotely, one can now simply implement a Flux workflow to handle this.

To perform remote process management, one can do the following:

- start a remote process in background mode and then,
- store the process id returned in the stdout of the RESULT from the Flux SSH Command Action in a workflow variable “pid”.

Here is an example remote command that the business would like to run, and ensure it completes within a specified duration.

```
Remote Command=nohup staging/generateReports.sh > out 2> err < /dev/null &
echo $!
```

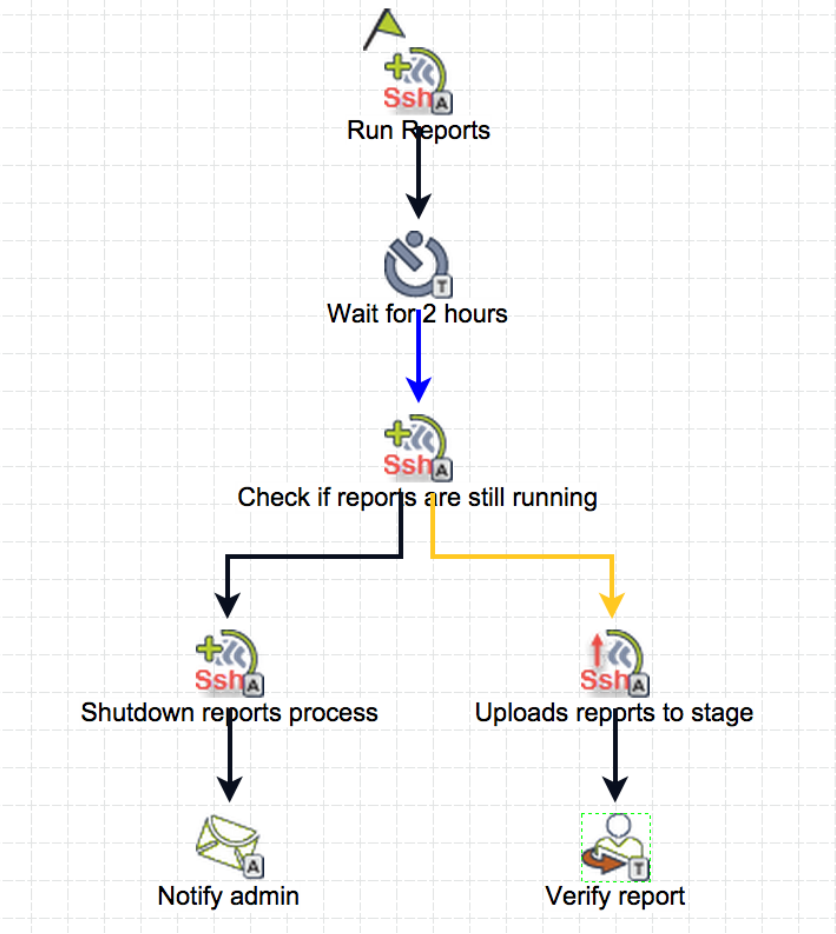
This command would return the process id of the running program which is then stored in the workflow variable “pid”. Assume it is expected that this program is to complete in 2 hours. A workflow is desired to check if the process is still running after that time. This workflow queries the process id of the generatereports script using another remote command, which simply returns the name of the process if it is still running or none if the script has finished executing:

```
Remote Command=ps -p ${pid} -o comm=
```

In normal circumstances, one would see the process finish executing and then initiating downstream processing further in the workflow, but if it is still running the workflow will shutdown the process and notify administrative personnel. Issuing a kill command as shown below within an SSH Command Action will shutdown the running process.

```
Remote Command=kill -9 ${pid}
```

An example Flux workflow illustrating this workflow is shown below.



In Summary

Flux's introduction of SSH File Upload, Download, and Command actions further extends the integration capabilities and control available with Flux workflows. These powerful actions interact cleanly in a secure

infrastructure to better control and more centrally manage the diverse and distributed processing available within the modern enterprise.

About Flux

Flux assists enterprises in provisioning, onboarding, scheduling, tracking, and reporting an enterprise's file orchestration processes. These orchestrations vary from simple file transfers to highly complex workflows involving extensive processing, many routes, varied alerts, and complex decisioning. First released in 2000, Flux has grown into a file orchestration and workflow scheduling platform that enterprises rely on daily for their mission critical systems.

Contact Flux

Contact sales at: sales@flux.ly
For further information browse: flux.ly