RunDeck Workflow step/flow control

Goals

1. Add flow-control steps in workflows that pause the normal workflow execution until some user input is received via the GUI(/api?). This allows operators to review the current execution state and output before allowing a process to continue, and optionally providing further input to the execution.
2. (secondary) Add if-then-else conditional steps within a workflow, allowing evaluation of some criteria, and then switching between several options based on the evaluation

Job workflows should allow any step to require this user input. User input can be of the form of simple flow control: “Continue”, “Skip Next”, “Fail”, “Succeed”. Or it could be similar to Job options, requiring form input of different kinds for further information about next steps.

Optional: some set of users/email addresses could be sent notification that approval/input is required for the execution to continue. This could tie in with ticketing systems or approval-based change control mechanisms.

If-then-else conditionals would require evaluation primitives: comparing two values and using that basis for the flow control. The compared values could be defined as property references to allow use of user input values already in the execution context (option values), e.g. “${option.example} == something”. Additionally, perhaps the conditional needs to know something of the current job execution state, and use the result status of a previous command step as part of the conditional, like “${step.previous.status}==failed”.

The user-input flow control could be built from the if-then-else flow control primitive; user input is simply added to execution context and then compared in the conditional.

Mechanism

Add new “command” primitives for workflows:

- User input step - defines a set of Options similarly to a Job, and can include an optional set of flow control options (continue, skip, fail, succeed) that the user can select from
    - the trivial case of no user input and only flow control options satisfies the approval/checkpoint
    - the case of user input with no flow control options is equivalent to “continue”
    - optionally, some kind of notification can be sent to indicate that user input is required for the execution

- conditional step - defines two values to compare at runtime and primitive comparison operator. defines flow control results if the condition is met: (continue, skip, fail, succeed)
- “else” conditional step - defines flow control action to take if previous condition is not satisfied: (continue,skip,fail,succeed)

Should existing “commands” be augmented with these input/conditional abilities, or should these primitives exist as their own steps in the workflow?

Multiple Nodes and Threads

Jobs can run with multiple threads across multiple Nodes.

For example, a job could be executed on 20 nodes with 5 threads.

In this case, how should user input be handled?

Should user input/approval requirements act as synchronization points whereby all Nodes must reach that point in the workflow prior to proceeding with a single user input?

Should each Node’s workflow be treated separately and require 20 possibly different user responses, one for each Node?

Ideally we would have something like the following:

A Job is executed on 20 nodes running on 5 threads. The workflow contains a step which requires user approval to continue. Some of the threads reach this step in the workflow, and possibly an email notification is sent out with a link to the rundeck execution page. The execution page now displays a status indicating that user input is required, and if the user is authorized to do it, shows the form input controls for the user input. The form contains a question like “Allow execution to proceed?” with options “Continue”,”Halt Execution”. As well, a list of Nodes that have reached this approval step are shown, with checkboxes. The user can select all/none or individual nodes. The user can also select a checkbox “Apply this answer to all remaining Nodes”. If the user applies the choice to only a subset of the nodes, those workflows will proceed following the control/approval response. The user approval/response user input box will go away if all nodes have been responded for, otherwise it will be refreshed or reappear later if further approval input is required.

User Interaction

User interaction for input while the execution is running requires changes to the job execution system, and job execution GUI. Previously all user input was performed prior to job execution.

Execution service will have to be updated:

- New state flag to indicate user input is required.
- In-memory state of control steps and nodes, to record which nodes require user input.
- Asynchronous user input system, allowing gui/api to send user input, and execution service to receive it