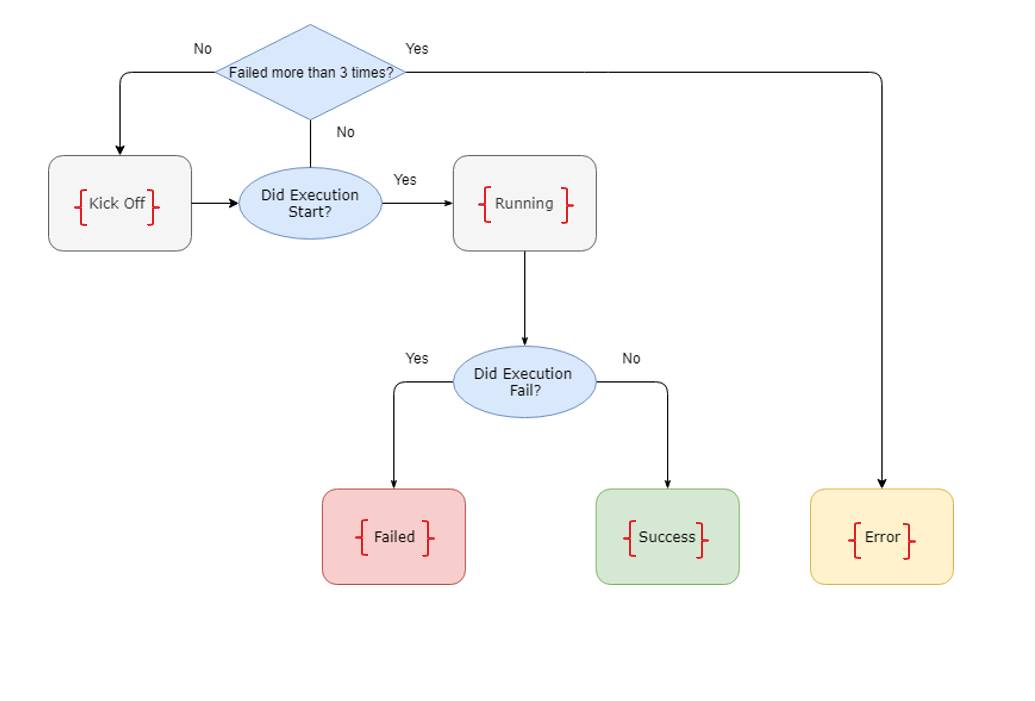
**EXECUTION AGENT**

Execution Agent is the component in the Execution Server ecosystem that is responsible for triggering of commands sent by the Execution Server, tracking them and sending their states to the StateServer.

Following is a high-level workflow of how the agent would fit in the Execution Server ecosystem:

1. Execution Server forks off the Execution Agent passing it the Tracking ID, Command to be executed and Command properties.
2. Execution Agent sends a READY state signal to the State Server notifying it has been initiated by the Execution Server.
3. Execution Agent interprets the Command Properties and batches of the commands in a sequential fashion.
4. Execution Agent tracks the forked off command and sends periodic RUNNING state notifications to the State Server.
5. After the commands are finished the Agent send a final state of SUCCESS or FAILURE.



IMPLEMENTATION STRATEGY:

The agent would be written in python to keep it as light as possible and scale as new use cases get onboarded. This would be a non-persistent utility that would be started per execution by the Execution Server.

The process forking and tracking would be done using the native python subprocess module or a 3rd party library like executor (<https://pypi.org/project/executor/>).

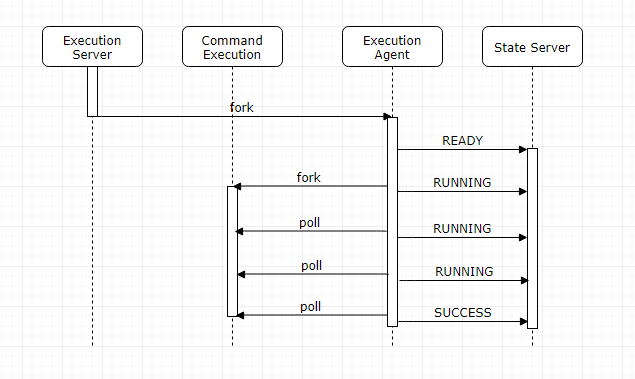
Execution Agent will send status updates to the State Server through BMS.

COMMAND PROPERTIES:

It is the execution agent that enforces the command properties on the actual execution. These can be:

* Retry Count
* Ulimit

WORKFLOW:



OPERATIONAL SCENARIOS:

Host Failure:

Execution Server follows a fire and forget model. If the host goes down all the executions on it are lost. State Server will have an internal mechanism to listen on state updates from Execution Agents. If we don’t get a status update within a specified timeout period State Server will notify using ProdMonitor and the rerun would have to be done manually.