TroubleChute

General flow of Node-Red:

1. Node-Red receives data from IoT Sensors in the form of a payload. IoT data is recorded and sent to the cloud at a fixed interval.
2. Data received from the IoT sensors are also being sent to the Dashboard in real time for active monitoring and visualisation of data. A new column "smoke_detected" has been generated in the Dashboard (indication of presence of variable "smoke"). A gauge for indicating temperature, a line graph to plot the variations in temperature, and a table have been included in the Dashboard for clear and concise conveying of information.
3. At the same time, data is sent down the pipeline with switch nodes to check the "smoke" variable in the payload. We have set thresholds representing fire as ("temperature" > 40 or "smoke" = 1)
    a. In the case of sensors indicating no fire
        i. No action is taken, with logs being recorded in the "nofire" database
    b. In the case of sensors indicating fire
        i. If smoke = 1, the set_count function will initiate and add a "count" variable to the payload. *(Refer to NOTE ON SET_COUNT to understand use of "count")*. "Count" variable is set to 1, and the Flush system is thus activated.
        ii. If smoke = 0, the check_temp switch node will check the "temperature" variable in the payload. If temperature > 40, the set_count function is also called, and the "Count" variable is set to 1, activating the flush system.

NOTE ON SET_COUNT:

The use of the set_count function is to enable our multi pronged logic by allowing us to identify messages being sent by the IoT sensors and thus sequence our increasing response to fires. Only when the fire gets smokey("smoke" = 1) /hot enough ("temperature">40) will the set_count function be initiated. The first message that fulfils this criteria is given the "count" value of 1, signalling the activation of the flushing mechanism. The second message that fulfils this criteria indicates that the flushing mechanism has not been sufficient thus requiring an escalation of response. This message is given the "count" value of 2 and signals the activation of CFRs through automatic MyResponder and Social Media (Twitter) postings. Subsequent messages will be given the "count" of 3, with no programmed response attached to the message. This is because after the first flush's failure, we are unsure if water is the most efficient/safest way of extinguishing the fire. We do not want to annoy/bombard the public with notifications as well.

4. After the flush, the IOT device will send in data in the next time interval, and if the fire is still indicated as present, CFRs will be activated to help the situation through automated MyResponder notifications and automated Social Media posts. We have managed to carry out automation of tweets through usage of api keys from twitter, allowing for the tweet node in node red to push out timely messages on twitter. This has been enriched through google maps query and javascript parsing to include location links to provide directions to the fire via google maps.
5. After CFRs have been informed, the message flows into a 5hr delay, providing the responders and SCDF time to deal with the fire and assess any property damage. After the delay, the message passes through a function node that resets the count variable, and thus resetting the whole node red system.
6. Throughout the entire flow, data from the sensors are constantly being stored in the “yesfire” and “nonfire” database created. Only storing of payload object has been enabled on all cloudant servers receiving sensor data to enable easier viewing in the database.