README for COMP 598 Project1

Group 4: Fardin Abdullah, Michael Stoski, Md Alauddin Siddiqui

Note:
Use cloud init to initialize everything before starting
run cloud init every time to make sure the 50 nodes are initialized again

Do not run cloud init twice in one session

Run it every time the proxy or the resource manager restarts so we can make sure the initial 50 nodes are present in the nodes list.

1. **cloud init** - This initializes the 50 nodes. This has to be run at first in order for every command to run later.

2. **cloud pod register <podname>** - This registers a new pod in the main resource cluster list. This is again just adding a Pod object and not useful for this assignment but will be used in future assignments.

3. **cloud pod rm <podname>** - This removes the specified pod from the main resource cluster. We cannot remove a non-existing pod and we also cannot remove the default pod.

4. **cloud register <nodename>** - Here, we only have one argument which is the node name. Since for this assignment we are only having one default pod, we are not putting any argument for pod id to avoid overcomplication.

5. **cloud rm <nodename>** - This removes the existing node from the default pod. We cannot remove any of the initial 50 nodes or a non-existing node.

6. **cloud launch <pathtojob>** - This function works as expected although we ran into significant difficulties running the actual contents of given sh files. The contents of the file are sent all the way to the proxy where a filler command used to demonstrate the launch functionality (sleep 30) is substituted in place of the job that should be run. The RM assigns jobs to Idle nodes and queues them if none are available. Using threading, multiple jobs (just sleep 30 currently) can be run at the same time in different containers,

which is visible from the dashboard. In the cloud dashboard, you can continuously refresh to see when jobs are running and once they become idle.

Dashboard URL: http://127.0.0.1:3000/cloud/dashboard/
or
Dashboard URL: [IP_ADDR_VM2]/cloud/dashboard/

7. **cloud abort <jobid>** - This function sends the jobid through the resource manager, to the proxy. Once arriving in the proxy, however, it does not (in our current implementation) abort the job associated with the given ID.

CLI COMMANDS

1. **cloud pod ls** - Lists all the pods in the resource cluster. This is like a dummy command since for this assignment 1, we actually only have one pod. After cloud init, the default pod gets created. We are assigning dummy variables for the id and nodes.

2. **cloud node ls** - For this command, since we don't have multiple pods, this just returns a list of nodes (with their information) for the default pod only.

3. **cloud job ls <node name>** - If used without the argument of node id, it lists all the jobs assigned to nodes. If a node name is given, it will list all of the jobs that have ever been assigned to this node. Note we are only returning and logging the contents of the job files currently. In future implementations it will also return status.

4. **cloud job log <jobid>** - We implemented the skeleton of this function with the API endpoints and the commands are invokable from the CLI, however, they do not currently perform correctly.

5. **cloud log node <nodeid>** - We implemented the skeleton of this function with the API endpoints and the commands are invokable from the CLI, however, they do not currently perform correctly.