Ben Rivera & Eddie Goode
Professor Dubey
CS 3281
4 May 2017

Linux Orchestration Framework Design Specification

Class Architecture:

- MasterServer.h/cpp
    - Private:
        - ServerPort #
        - Client struct (map(pids, status), alive, socket)
        - Client vector
        - numClients
    - Public:
        - MasterServer()
        - ~MasterServer()
        - testConnections() //heartbeat message
        - start()
            - infinite loop:
                - establish tcp connection
                - read in user input
                - send heartbeat message
                - loop through each client
                    - parse command
                    - send command to client
- Commands:
    - copy – copy followed by file
        - use scp and get the cwd of both master and client node
        - request cwd of clients using socket
        - system call scp command
    - list PIDs – pid
        - First, update status of list
            - This is done by sending pid to client with pid number and calling waitpid with WNOHANGE which will show whether it is running or if it has stopped and why
            - Have client send this information back through socket
            - Output list with
                - PID
                - Status
    - List clients – list
        - This will be accomplished by simply printing out if alive
    - SIGKILL – 'sKill" followed by pid

        - This will be accomplished sending sKill and process to client
        - Client then uses the system kill(PID, SIGNALKILL)
    - SIGINT – ‘kill” followed by pid
        - This will be accomplished sending kill and process to client
        - Client then uses the system kill(PID, SIGNALKILL)
    - Exit – exits loop and allows client to connect
    - Execute – ‘./’ followed by file
        - Send buffer to client
        - Client parses buffer and uses app exec command
        - Client will then call waitpid on process to get status
        - Client will return pid and processes current status
        - Master will add pid and current status to the correct worker nodes map with processes and status
    - Remove – rm followed by file
        - This will remove the file by sending to apps execute command
- App.h/cpp
    - Use capabilities of app class to execute commands and create new processes
    - Execute() forks a new process, calls dup2() to log file, and call execvp on command

- MasterServerMain.cpp
    - Main() //run start()

- ClientServer.h/cpp
    - Private:
        - Serverport #
        - Buffer size #
        - getChildStatus(pid) //for status functionality
    - public:
        - serverClient()
        - ~serverClient()
        - start()
            - infinite loop:
                - establish tcp connection
                - recv command
                - check for implemented command
                - create app and execute
                - Command function described above
- ClientServerMain.cpp
    - Main() //run start()
- PracticalSocket.h/cpp
    - Use practical socket class to establish tcp connections.
- Utilities.h/cpp

    - Use parse functions for command parsing
- CMakeLists.txt
    - Use to build client/master executable

Integration and Testing:

1. We will first establish connection from Master node to client nodes
2. Once we are sure connection is established and secure, we begin adding functionality.
3. Begin with copy and add each functionality one by one testing for each with multiple clients
4. Each command is added on the corresponding master and client side together

Assumptions:

1. We are assuming that the master wants to perform all actions on all nodes because it is a deployment framework. However, we must keep track of which clients have which processes for the kill commands and list pid commands
    a. This will be accomplished by adding a std::map children to our struct which each client has
2. We assume that the proper SSH connections are already established and do not have to be written in using the program because that is beyond the scope of this project.
3. We are assuming TCP connection because it is a stateful application