LINUX ORCHESTRATION FRAMEWORK

Ben Rivera & Eddie Goode

CS 3281

4 May 2017

PROJECT DESCRIPTION

- An orchestration infrastructure is responsible for application installation and lifecycle management of apps on a collection of remote nodes
- Capable of launching & shutting down apps on remote nodes
- Each remote process writes to a log file that can be returned to master
- User can query each node's CPU & memory usage

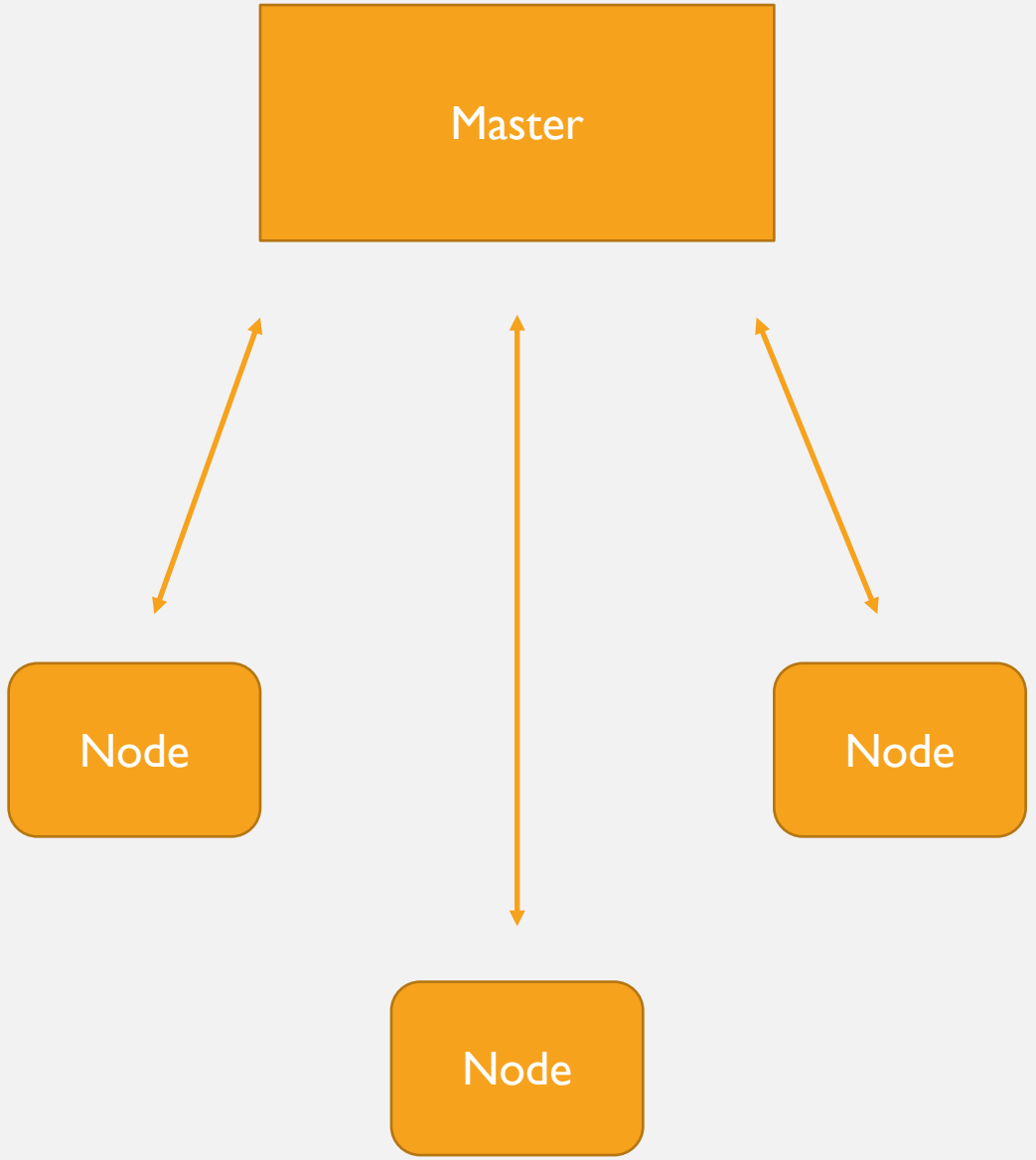
PROJECT ARCHITECTURE

Master:

- Running a continuous prompt for user input
- Connects to nodes with TCP sockets
- Issues ‘heartbeat’ messages to every node
- Maintains a list of active nodes

Node:

- Executes commands using fork() and exec()
- Utilizes app() class for parallel execution
- Logs output and errors to log.txt file using dup2



CHALLENGES

- Sending messages across TCP sockets using a buffer. We had trouble sending a complete message and often had random trailing characters at the end of a transmission
- Deciding how to utilize the functions of the app() class while also maintaining a list of PIDs
- Correctly using dup2() to create a log file and sending output to the file
- Maintaining a list of actives nodes

WHAT WE LEARNED

- How computers can communicate effectively using TCP sockets
- How to dup() file output to a new fd
- How to create and use ssh keys
- How to use git effectively when working in a team

WORK DISTRIBUTION

- We both worked side-by-side for most of the project. If there was ever any trouble, we asked each other for help. Eddie and Ben both contributed to writing , testing, and documenting the project.