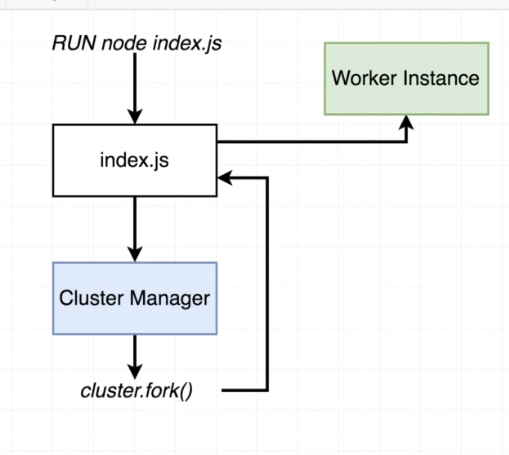
**NodeJS Theory**

1. The NodeJS project is an API interface to the V8 and libuv dependencies. V8 purpose is to allow to execute JS code outside of the browser. Libuv allows us to write code in JS and it is translated to C++ so it can provide access to the files system, networking, concurency etc.
2. V8 is 70% C++ and libuv is 100% C++ code.
3. Node js has Standard library modules integrated (http, fs, crypto, path…)
4. Diagram

   Description automatically generated
5. V8 has the role to translate the JS Code (Integer, Exception, String…) to the C++ equivalent.
6. Threads are used to process some instructions and are waiting to be executed by the CPU. Each process can have multiple threads. The system control the scheduling of the threads to speed up the execution. The system can process more threads if it has multiple cores, or if the system can detect that some operations take longer time and it can execute other threads in between.
7. Node Event Loop is single Threaded, but Some of Node Frameword/Std Lib are NOT single Threaded.
8. The node is NOT single threadedDiagram

   Description automatically generated
9. The V8 delegate some “expensive” tasks to the node C++ side that have a thread pool with 4 default threads.
10. Can we use the threadpool for javascript code or can only nodeJS functions use it? We can write custom JS that uses the thread pool.
11. What functions in node std library use the threadpool? All “fs” module functions. Some crypto stuff. Depends on OS (windows vs unix based).
12. How does this threadpool stuff fit into the event loop? Tasks running in the threadpool are the “pendingOperations” in our code example.
13. For requests and async code that require access to low resources like networking(requests) the UV has libraries and functions to let the OS to handle the requests, and it is no longer limited to the thread pool, it is limited to the OS threads. (OS makes the fetch request).
14. What functions in node std library use the OS”s async features? Almost everything around networking for all OS’s. Some other stuff is OS specific.
15. How does this os async stuff fit into the event loop? Tasks using the underlying OS are reflected in our “pendingOSTasks” array. This can be network requests, app.listen(“port”)…
16. Summary Diagram

    Description automatically generated
17. The HTTP request don’t touch the thread pool, it works with the OS resources directly.
18. Node internal threadpool and HTTP Requests that work through OS network resources. Diagram

    Description automatically generated
19. 
20. Clusters. First instance is working normally, but the if we call cluster.fork it will start another instance that have the Event Loop(So we will not have only one instance).
21. Clusters are greate to enhance the performance. One tool that is needed in production environments is the pm2.
22. JSON.stringify, JSON.parse to store objects in Redis cache
23. To not use callbacks, we can Promisify the function to return a promise, and then to use the promise.
24. Testing challenges: Need to somehow laungh Chromium programatically and interact with it from a test suite. How do we make assertions in jest about stuff that’s happening in a chrome window? How do we “simulte” loggin in as a user? We’re going through Goolgle OAuth.