```

import { WorkerClass } from './workerProxy_2.js';
import { Job } from './Job_1.js';

///////////////////////////////////////////////////////////////////////////////
//
// 整合所有的 worker
//
///////////////////////////////////////////////////////////////////////////////

class WorkerManager {

    static get_instance(workerCommand) {
        if (WorkerManager.instance == null) {
            WorkerManager.instance = new WorkerManager(workerCommand);
        }
        return WorkerManager.instance;
    }
    //----------------------------------------------------------------
    constructor(root) {

        // 與外界橋接的橋樑
        this.root = root;

        //
        this._;

        this.settings;

        this.environment;

        // 記錄有多少在線的 worker
        this.workers = new Set();

        // 正在閒置中的 workers(等死中)
        // this.idleWorks = new Set();
        this.jobList = [];

        // 因應各種環境引入不同的 worker
        // 一個重要的設計點
        this.workerClass;

        //-----------------------
        this._getSettings();
    }
    //----------------------------------------------------------------
    _getSettings() {
        debugger;

        this.settings = this.root.settings;

        this._ = this.root._;

        if (this._ == null) {
            throw new Error('no import _');
            // console.log('no connection with _');
        }

        if (this._.$extension1 == null) {
            throw new Error("need import _extension1");
        }

        // 取得環境
        this.environment = this._.$extension1.info.environment;

        //-------------------------
        // 取得適合當前環境下的 workerClass

        switch (this.environment) {
            case "nodejs":
                this.workerClass = WorkerClass.NodeJsWorkerProxy;
                break;
            default:
                this.workerClass = WorkerClass.WebWorkerProxy;
                break;
        }
    }
    //--------------------------------------------------------------------
    addJob(args) {
        // debugger;
        // console.log('--------------');
        console.log('WorkerManager > 加入新工作');

        let job = new Job(this, args);

        // 把任務加入
        this.jobList.unshift(job);
        //----------------------------
        // 檢查是否有 idle worker
        this.noticeWorkers_checkJob();

        return job.promise();
    };
    //--------------------------------------------------------------------
    // 預先不會在一開始啟動 workers
    // 通常只有在有指令後才會有 workers 待命
    // 不過可以事先就請 workers 待命
    initWorkers(count) {
        debugger;

        const min_workers = this.settings.min_workers;
        const max_workers = this.settings.max_workers;

        if(count > max_workers){
            throw new Error(`initWorkers.count > max_workers(${max_workers})`);
        }

        for (let i = 0; i < count; i++) {
            debugger;

            if(this.workers.size >= max_workers){
                break;
            }

            let employment = false;
            if (this.workers.size < min_workers) {
                // 正職還有缺額
                employment = true;
            }
            new this.workerClass(this, employment);
        }
    }
    //--------------------------------------------------------------------
    // 針對 node.js
    terminateAllWorkers(){

    }
    //----------------------------------------------------------------
    // 請工作人員注意是否有工作進來
    // worker: {worker: 由 worker 呼叫，null: 由 manager 呼叫}
    noticeWorkers_checkJob(worker) {
        if (worker == null) {
            // console.log('check by manager');
        } else {
            console.log('check by worker(%s)', worker.id);
        }


        while (this.jobList.length > 0) {

            console.log('still have jobs(%s)', this.jobList.length);

            // 盡可能招募 worker 來接工作
            let w = this._checkIdleWorker();

            if (w) {
                let job = this.jobList.pop();
                w.takeJob_callByManager(job);
            } else {
                // 若找不到可用的 worker 作罷
                break;
            }
        }


    }
    //----------------------------------------------------------------
    // 新增 worker(由 worker 自己通報)
    addWorker(workerProxy) {
        this.workers.add(workerProxy);
    }
    //----------------------------------------------------------------
    // 移除指定的 worker(由 worker 自己通報)
    removeWorker(workerProxy) {
        this.workers.delete(workerProxy);
    }
    //----------------------------------------------------------------
    // worker 想取得 job
    getJob_callByWorker = function () {
        let job = null;

        console.log('有(%s)項工作待領', this.jobList.length);

        if (this.jobList.length > 0) {
            job = this.jobList.pop();
        }

        return job;
    }
    //----------------------------------------------------------------
    // 檢查是否有空閒的 worker
    _checkIdleWorker() {
        // debugger;

        console.log('manager find worker');

        const max_workers = this.settings.max_workers;
        const min_workers = this.settings.min_workers;

        let idleWork;
        // 找尋空嫌中的 worker
        let idleWorks = this.findIdleWorkers();

        if (idleWorks.length > 0) {

            // 優先找正職者
            idleWorks.some(function (w) {
                if (w.employment) {
                    idleWork = w;
                    return true;
                }
            });

            idleWork = idleWork || idleWorks[0];

            // console.log(`manager find idle worker(${idleWork.id})`);

            return idleWork;
        }
        //----------------------------

        // 沒有空閒中的 worker
        if (this.workers.size < max_workers) {
            // 沒有閒置的 worker
            // 但已有的 worker 數量尚未達上限

            let employment = false;

            if (this.workers.size < min_workers) {
                // 正職還有缺額
                employment = true;
            }

            // console.log(`manager employment new worker(employment: ${employment})`);

            new this.workerClass(this, employment);

        } else {
            console.log('manager no find worker');
        }
    }
    //-----------------------------------------------------------------
    // 找出閒置中的 worker
    findIdleWorkers() {
        let workers = Array.from(this.workers);

        workers = workers.slice();

        workers = workers.filter(function (w) {
            if (w.isReady2TakeJob()) {
                return true;
            }
        });
        return workers;
    }
    //-----------------------------------------------------------------
    // 取得需要的資訊
    getAllworkersInfo() {
        let all = [];
        let idle = [];
        this.workers.forEach(function (w) {
            all.push(w.id);

            if (!w.isBusy()) {
                idle.push(w.id);
            }
        });

        let jobCount = this.jobList.length;
        //----------------
        return {
            all: all,
            idle: idle,
            jobCount: jobCount,
        }
    }
    //------------------------------------------------------------------
}

WorkerManager.instance;

//===========================================================================
WorkerClass.GModules["WorkerManager"] = WorkerManager;
Job.GModules["WorkerManager"] = WorkerManager;

WorkerManager.GModules = {};

export { WorkerManager };
```