

IOE

IntelligenT Operating System Exchange

內容

摘要	3
簡介	4
系統架構與節點 Node	6
CryptoNode	7
IaaSNode	9
JenkinsNode	9
DockerNode	10
交易所 Node	11
IOE Client Kit	11
IOE 瀏覽器插件	12
代幣發行細節	13
IOE 代幣	13
發行方式	13
分潤費與分潤紅利	13
開發費與使用目的	14
實作時程	14
代幣發行 ICO 與推廣	14
應用開發	15
第三方人員開發	錯誤! 尚未定義書籤。
轉為公鏈	15
引用的項目	16

摘要

在本白皮書裡，我們將介紹 IOE 去中心運行架構，它是一個去中心化的雲端運行解決方案，透過公開和可監督的雲端運行架構來達到虛擬物品有價化、交易安全且公正。去中心化的核心想法就是應用程式是可以被監督，讓虛擬的物品能夠有所保障，但是目前的解決方案都不行與現行雲端架構做整合，本白書就來解決這個問題。

虛擬物品有價化其實目前最能夠實際應用的就是遊戲中的虛擬寶物，且從網路遊戲發跡以來，虛擬物品交易是一件平常的事，但是有這個需求虛擬物品的價格通常都是極度浮動的，而且通常只跌不漲，那這要歸咎於遊戲的生態，有以下幾點

- 物品的稀有程度是由開發商所決定的
- 沒有一個安全公正的流通管道
- 開發商為了營利大量販售虛擬物品
- 開發商為了吸引用戶的手段，就是大量贈送虛擬物品

以上幾點導致虛擬物品沒有穩定的價格，導致收藏的價值低落。

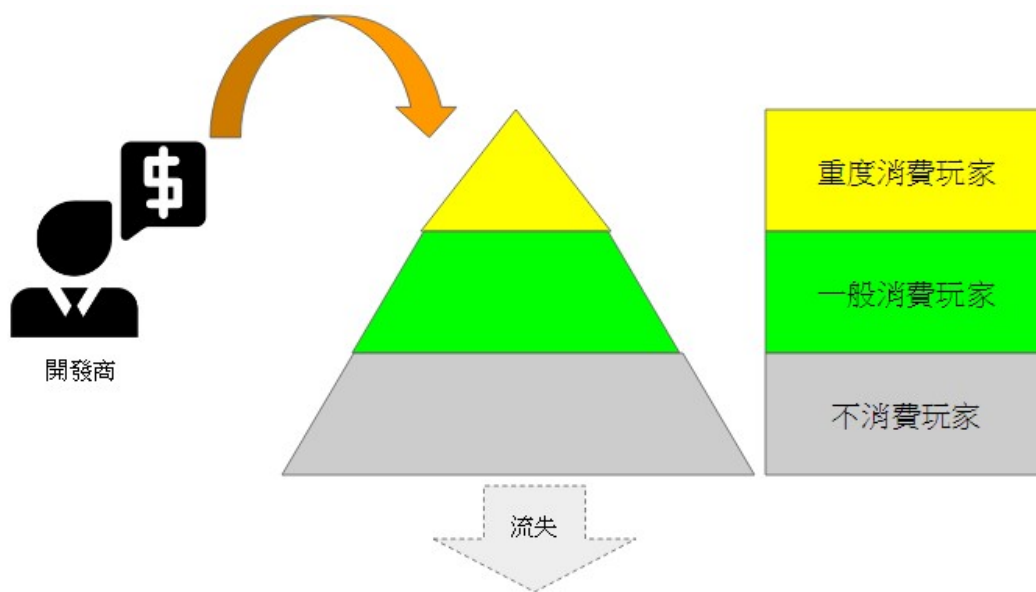
所以本架構，透過去中心化的想法，讓開發商的程式碼透明，供用戶檢視，並且虛擬物品寫入在我們的加密節點中(CryptoNode)中，讓虛擬物品可以看發行的狀態，並且無法篡改，讓它保有像是真實物品的稀缺度，再來運行的後台伺服器透過我們的 laasNode 運行節點，可以讓運行的程式碼一定程度透明，已確保虛擬物品不會被惡意操作或是黑箱處理，綜合上述幾點讓虛擬物品具備保值的商品該有的條件：公開，透明，稀缺性等屬性。

那本系統分成兩種節點分別是：laasNode 和 CryptoNode, laasNode 主要是負責去中心化的運行環境，確認運行程式的透明，且不可以修改。CryptoNode 是負責敏感資料，這邊的資料都會有加密，並且無法串改，只有對應的 laasNode 可以做操作和修改。

簡介

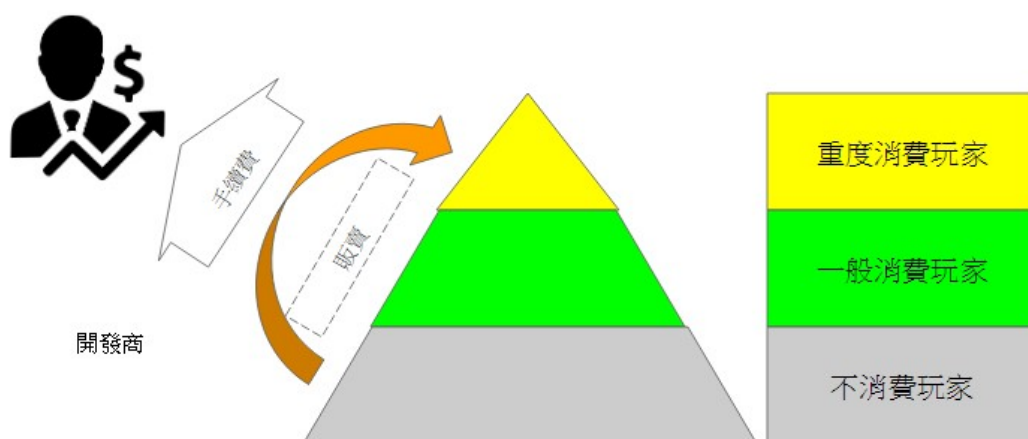
區塊鏈技術中的 dApp (1),其實最有用的就是虛擬物品現實化,而虛擬物品目前最直接的就是遊戲中的虛擬點數和虛擬寶物,並且遊戲的全球市場 1370 億美金以上 (2),所以用遊戲產業來推動區塊鏈技術是最為可行的,任何科技技術要商業化,基本上娛樂業是功不可沒,像是雲端技術的普及,Facebook是功不可沒的 (3),更別說 Facebook 當年也是靠小遊戲捕獲大眾市場 (4),本團隊透過區塊鏈去中心化技術來解決遊戲產業的問題。

目前遊戲產業的成長都會遇到四個階段 (5),而且近年的成熟期和衰退期越來越快,有以下幾種問題造成,第一開發商過快開發新的版本,導致能習慣遊戲的操作甚至對此遊戲上癮的玩家流失 (6),那麼不改版又無法吸引新的玩家,因為玩家是受外部社交軟體所影響所以需要積極的改版來吸引玩家,不然開發商會無法獲利,變成明明已經進入成熟期的遊戲卻還是要如成長期的遊戲改版,導致遊戲快速到衰退期 (7),第二點就是遊戲課金導致休閒玩家的離開 (8),並影響整個遊戲生態系(圖表 1),只要太過於販售遊戲物品就會有殺雞取卵的效果,如果沒有花費的玩家就無法繼續玩遊戲,使得不消費的玩家就離開了。



圖表 1 目前課金遊戲生態金字塔

那如果能夠能讓遊戲玩家組成一個循環的生態系,如圖(圖表 2),那就能夠讓遊戲內的用戶,能夠把自己獲得的虛擬寶物,透過一個公開的交易所交易,並且的這個交易所是能夠是有價的代幣購買賣的,這樣官方就不用擔心虛擬寶物的通貨膨脹,反而虛擬物品設計的較稀有,反而會有更大的利潤,這樣不消費的玩家也可以透過自己的時間來換來相對的利潤,那花錢的玩家可以省下時間,這樣對雙方玩家取得了公平的平衡。



圖表 2 自由交易生態

那我們要如何確定應用項目方是公正操作這些虛擬物品，以確定這些虛擬物品是公開、公正、不被濫用的，那這就我們架構要解決的辦法，在處理虛擬物品來說背後都會有一個伺服器來做處理，那麼只要能夠讓運行的架構和程式碼透明，不可以被修改，這兩點就可以基本確定這是一個公正的後台伺服器，那麼本架構來解決第一點架構和程式碼透明，透過 git (9)來把程式碼公開和檢視，再來透過 Docker (10)就可以把系統架構給公開了，那麼第二點不可被修改就用到我們 IaasNode(IaasNode)，這其實是一特製 Linux (11) 映像檔，只要用這特製運行的作業系統，就無法對該作業系統進行修改與調整，基本架構這樣就可以做到公正的後台，那麼玩家的虛擬物品的資訊可以透過[CryptoNode]來記錄這些有價的資料，這就是本架構的基本設定。

TODO 交易所 NODE

那麼虛擬物品交易需要一個有價的代幣，這個代幣就是本架構的(代幣)，本代幣是透過 ethereum (12)中的 ERC20 (13)的規範發行的區塊鏈代幣，現在市面上的也很多 ERC20 的代幣，但是並沒有實質的價值擔保，通常都會變為空氣幣 (14)居多，而本團隊是透過 Solidity 智能合約 (15)來實現透過 ETH 做為擔保與交換，那 ETH 中的 Ether (16)與 IOE 的匯率計算是透過方程式

$$Ether = (12.24744871 \times IOE)^2 / 10^{18}$$

來做匯率計算的，透過這個匯率可以向智

能合約做購買與販賣，由此可以知道當 IOE 購買越多其 Eth 的價錢也會變高，反之賣的越多價前就會下降，透過這種方式可以達到實質擔保，也可以透過合約交易來符合自由市場的供需與價錢的反應 (17)。

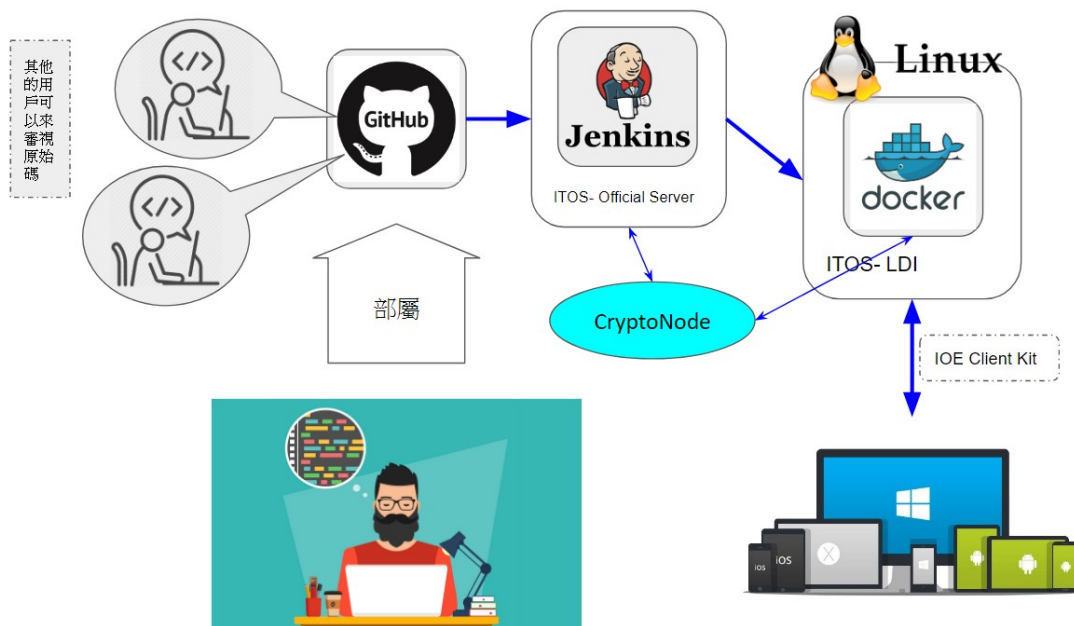
TODO 我們要先找應用

本團隊是要打造真正可運行的去中心化架構，因為目前市面上的去中心的架構都沒有實際的應用，反而多為投機的項目居多 (18)，所以本團隊會以目標導向為目的，想來找尋應用或是先實作應用為首要目標，

系統架構與節點 Node

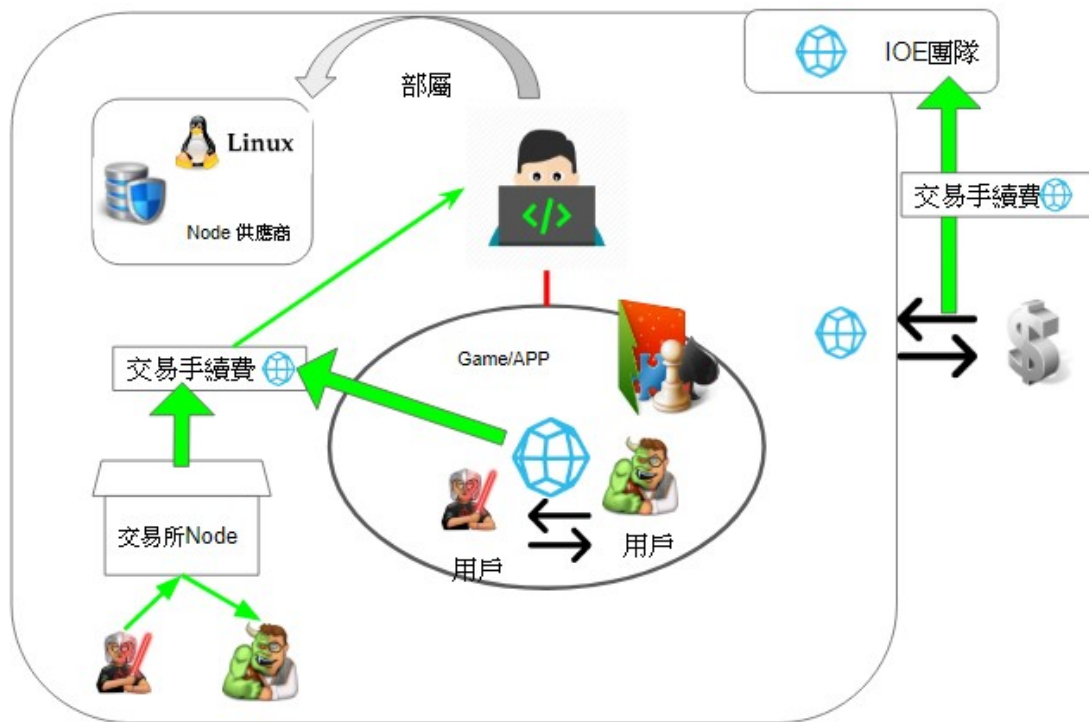
本系統的核心有兩個，第一把程式碼和應用所用到的架構可讓其他用戶檢視並且可以運行，第二能夠把有價或是敏感資訊放到去中心不能被串改的資料庫 [CryptoNode]，能夠操控[CryptoNode]只有審核過的程式才能呼叫，。

系統架構設計如(圖表 3)，本架構透過 Docker (10)技術中的特性，能把運行所需要用到的基礎建設已程式碼的方式記錄到 Dockerfile (19)，這樣程式碼和運行環境就能夠一起檢視，接者把程式碼和 Dockerfile 透過版本控制技術 Git (9)上傳到 Github (20)、Bitbucket (21)或是其他的開源網站，然用遊戲開發商選定要部屬到哪一個[IaasNode]，然後透過[JenkinsNode]完成部署的動作，然後部屬的 Iaas 和 git 位置關聯資訊是紀錄在[CryptoNode]裡，並且只有認可的[IaasNode]才可以透過[錯誤! 找不到參照來源。]操作[CryptoNode]裡的資料。



圖表 3 系統基本架構

為了支撐這個架構，我們把系統裡的角色分成四種：開發商、IOE 團隊、Node 供應商、用戶等 4 個角色，角色的關係如(圖表 4)，當用戶透過智能合約 (22)，來買賣就會幫助我們獲得一部分的 IOE,當作[錯誤! 找不到參照來源。]，而用戶玩家可以透過取得到的 IOE 代幣去系統內的[錯誤! 找不到參照來源。]，來購買自身所需的虛擬物品，反之用戶也可以把自身的虛擬物品透過此管道販賣出去，來換到對應的 IOE 代幣，而每種虛擬物品都是有屬於某個系統內的應用，如果該應用的虛擬物品被買賣，其中有部分的費用會給該應用的開發者，當作開發應用的報酬，當然開發商也可以直接跟用戶收取費用。



圖表 4 角色 IOE 的流動關係

CryptoNode

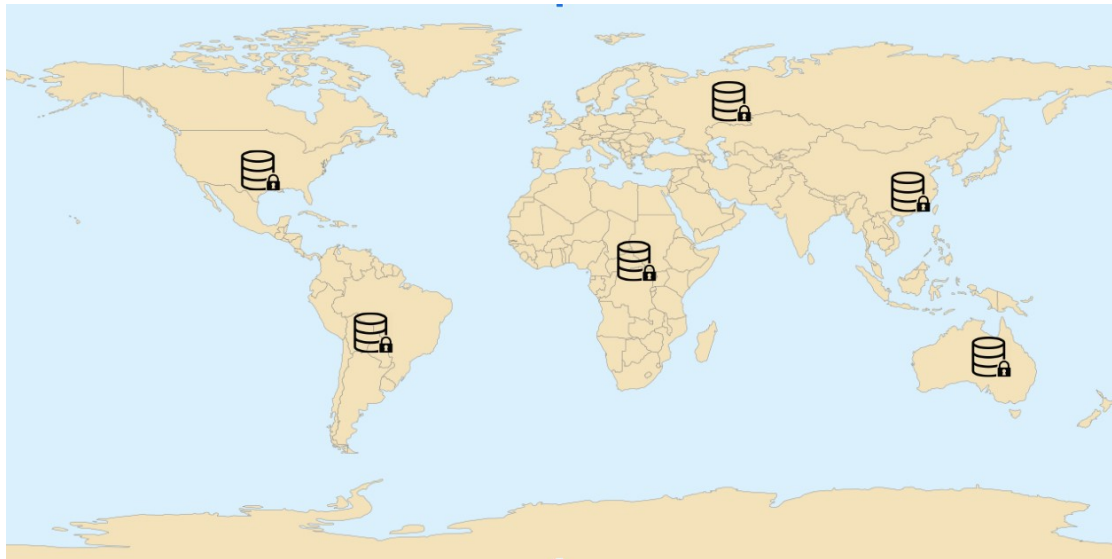
這個 Node 主要是紀錄本系統的用戶資料、App 資料、系統資料、JenkinsNode 資料、DockerNode 資料，要操作裡面的資料是要透過[錯誤! 找不到參照來源。]來操作和讀取，並請所以資料都有做加密處理，以確保資料的安全性。

這個 Node 是整個系統的核心，這也是最後才會變成公鏈的節點，並且 CryptoNode 並不支援挖礦，而是用戶來投票決定每個區域的 CryptoNode 供應商(圖表 5)，而當 CryptoNode 節點的供應商，依處理[錯誤! 找不到參照來源。]的交易量來獲得應有的手續費，這樣設計有兩種好處，第一這樣有去中心的效果，如果沒有只有一個節點被惡意串改，其他的節點也無法承認，第二點每區域都有節點就可以節省網路連線時間，這樣用戶就有更快的交易體驗。

CryptoNode 的確認方式和廣播方式是採用符合使用者行為去設計的，應用程式的所在位置通常都在固定區域，當應用程式發出[錯誤! 找不到參照來源。]的請求，只要該節點確認過該請求就可以回復給發出請求的 App，然後在發通知給其他的節點，讓他們更新資料。但是如果客戶如果原本都是美洲使用，現在出現在亞洲使用者的話，但是資料沒有同步完成就要等待同步完成才能繼續使用，但是正常情況這是不可能發生的。

CryptoNode 裡的紀錄[錯誤! 找不到參照來源。]，這是紀錄 IOE 團隊開發的 Jenkins (23)伺服器,當然這個伺服器運行在[laasNode]上，那這裡我們稱為

[JenkinsNode]，每[JenkinsNode]都對應多個可以部屬 App 的[DockerNode]用來，這些對應資訊也會記錄在 CryptoNode。



圖表 5 CryptoNode 分散式

CryptoNode 的用戶資料用(圖表 6)來做說明，每一筆用戶都有一組“用戶 ID”做索引，用戶會記錄用的基本資料，每次 CryptoNode 會新增修改不同的資料，其中“apps”是紀錄應用程式與用戶的對應資料，都會記錄在這裡，那如果有“@”做前綴就是系統有規範的資料格式，像是可以做交易的資料就會有規範。

```
{
  "用戶 ID": {
    "Name": "用戶顯示名稱"
    "用戶資料 1": "TEST"
    "apps": {
      "appId1": {
        "APP 資料": "xxxx"
      },
      "@IRC30s": {
        "irc30Id": 11
      }
    }
  }
}
```

圖表 6 用戶資料格式

IaaSNode

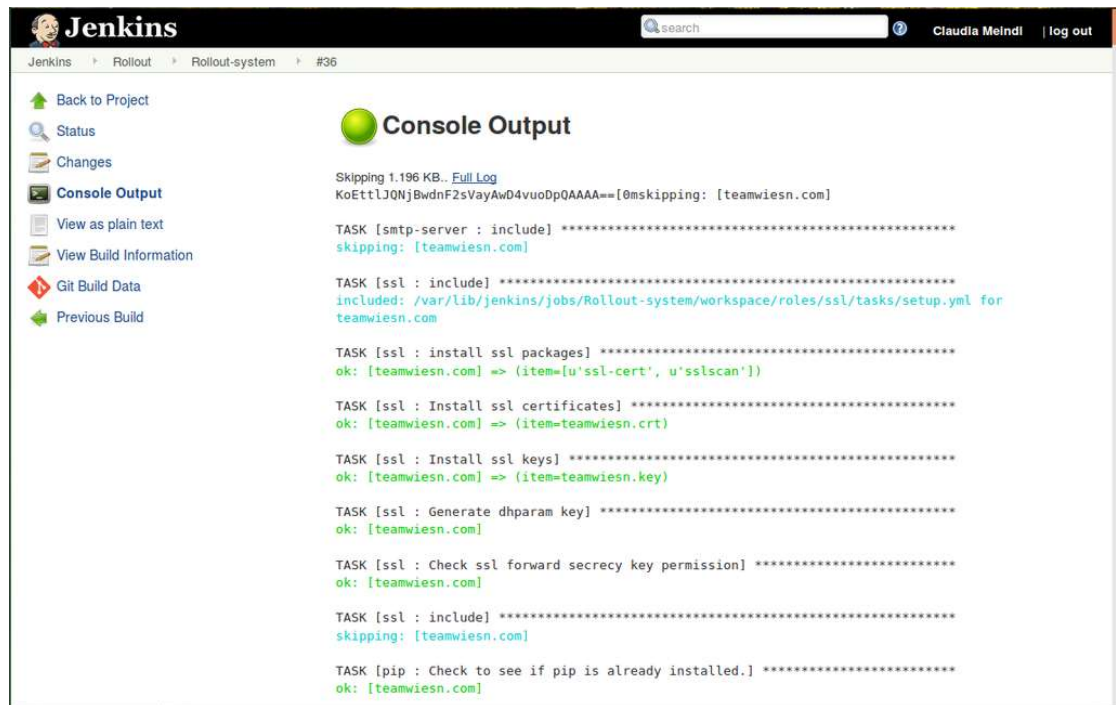
IaaSNode 是一種 Linux (11)作業系統 (24)的運行環境，它是由 IOE 團隊釋出的 Linux Image (25)所安裝後所運行的作業系統，此作業系統有三種特性，第一它只能運行特定的軟體或是應用程式，第二它透過任何方法登入，因為要確定無法修改或安裝其他應用程式，已確保由[JenkinsNode]部署過來的應用程式或是內建的應用程式不會被串改，只有允許的通道能能夠與操做這個節點，第三這種節點會有安裝[錯誤! 找不到參照來源。]的 SDK，已確保呼叫 API 是經過認證的 IaaSNode，並且每次傳送資訊前都會比對部屬程式的 SHA-2 (26)，已確保系統與 App 都沒有被串改。

JenkinsNode

JenkinsNode 也是運行在[IaaSNode]的其中一種，只是它上面是運行 IOE 團隊所修改的 Jenkins (23)伺服器，它是一個將 Git (9) 版本庫 (27)與運行的 [DockerNode]做連結，所以 JenkinsNode 的角色非常重要，所以這個節點會跟 [CryptoNode]是最後才會釋出。每個 JenkinsNode 所在位置等資訊都會記錄在[錯誤! 找不到參照來源。]中，開發者用戶只要透過瀏覽器連接到最近的 JenkinsNode，並且使用[IOE]，就可以登入 Jenkins 的後台(圖表 7)，開發者就可以指定自身專案的 Git 版本庫部屬到單一或多個[DockerNode]，開發商可以看到 DockerNode 列表，上面會有計價方式，此計價方式是申請 DockerNode 的節點供應商所決定的，開發商就可以依自己的考量來決定要來租借符合需求的節點。

上述是指第一次部屬應用程式的流程，之後如果要更動版本，首先開發商發送新的 Git Push (28)，這時候 JenkinsNode 就會收到這個新的版本，但是不會馬上部屬到 DockerNode，而是會發起投票機制，這時候如果完成投票，並且審核通過才會部屬到對應的 DockerNode，完成更新版本的動作。

投票機制的方式目前規劃是使用該應用程式的用戶，在一定的時間內要去 JenkinsNode 的後台投下是否允許更新版本，只要超過時間就算是廢票



圖表 7 Jenkins 後台

DockerNode

目前後台的組成有很多種，像是 Java、Nodejs、PHP、Python、Ruby、Go、C# (29)等等的程式語言編寫而成的，而且可能會需要用到很多第三方服務像是 AWS、Google Cloud (30)，更不用說需要安裝一些套裝軟體，DockerNode 的核心概念就是開發商不需要改變原本的架構，只要把交易的部分的程式碼公開，其餘開發商要用什麼架構都是可以的。

IOE 採用 Docker (19)的解決方案，能夠公開程式碼並且能夠檢視運行架構，且只要運行架構有支援 Docker 都可以使用，所以此節點才會命名為 DockerNode，此節點是一種 IaasNode，所以這個作業系統主要是運行 Docker，且只有預留通道給[JenkinsNode]。

DockerNode 會是最先開法讓用戶註冊的節點，用戶只要運行官方提供 Linux Image (25)，此時節點供應商要透過瀏覽器連接到最近的 JenkinsNode，並且使用 [IOE]，就可以登入 Jenkins 的後台(圖表 7)，並且輸入自己的節點網路位置，然後開始輸入想要租借的價碼，只要有開發商租借此節點就算是租借完成。而開發商可以透過(表格 1)的參數來決定是否要租這個節點。

參數名稱	單位	說明
出借價錢	IOE/小時	供應商決定
保證時數	小時	供應商決定，沒到運行到這個時數將會退”保證時數押金”
保證時數押金	IOE	供應商決定
關機提示	小時	供應商決定，要關機前要多久發出通知
關機提示押金	IOE	若沒有提示就關機，要給開發商這個押金
系統資訊	各種資訊	系統偵測，像是 CPU,Ram,Net 等等

表格 1 DockerNode 參數表

交易所 Node

只要開發商在[CryptoNode]中的應用程式資料中符合(表格 2)的格式規範，這些物品，這樣的物品就可以刊登在 IOE 虛擬物品交易所上，而這個交易所是 IOE 團隊所以開發的一個交易平台的節點，也只有它能夠被授權交易[CryptoNode]中的這些資料。

要交易的用戶可以過網頁瀏覽器可以進到交易所的網頁介面，並且需要搭配 [IOE]，用戶透過交易所就可以買賣虛擬物品。

規範名稱	名稱	說明
IRC30	代幣、個數用品規範	遊戲內的貨幣、消耗品，每個物品都是一樣的參數
IRC31	物品規範	例如遊戲中的裝備，每個物品可能有等級、顏色等參數的不同
IRC32	組合物品規範	這個物品是透過組合的物品
IRC40	人物規範	像是遊戲中的人物，寵物等等

表格 2 資料規範

IOE Client Kit

如果開發商要寫入或讀取客戶資料時須要透過 IOE 團隊開發的客戶端套件，因運行平台不同 IOE 團隊會提供不同的套件，目前規劃會釋放出三個程式語言的套件 C# Unity (31)、Javascript 、Java。用這些套件，確定用戶資料是安全不會被竄改的。

IOE 瀏覽器插件

由 IOE 團隊開發的瀏覽器插件，它類似於以太坊中的 **MetaMask (32)**，可以當虛擬錢包，並且可以與前端 **Javascript Client Kit** 做溝通，來用戶的資料傳輸更安全與便利。

代幣發行細節

IOE 代幣

IOE 代幣為 Intelligent Operating System Exchange Token，簡稱“IOE 幣”，代幣縮寫為“IOE”，發行總量恆定為 1 億個，且保證永不增發。IOE 幣是基於以太坊 Ethereum 發行的去中心化的區塊鏈數字資產，它是基於以太坊區塊鏈的 ERC 20 標準代幣。

發行方式

與透過 IOE 的 Ethereum 智能合約 (33)透過 Eth 做為擔保與交換，ETH 中的 Ether (16)與 IOE 幣的匯率計算是透過方程式(方程式 1)來做匯率計算，可以透過 IOE 的 dApp 網頁 (34)，來進行購買或是買賣，不過這並不包含系統開發費與分潤費。有關詳細的操方可以參考官方的教學說明 (35)。

$$Ether = (12.24744871 \times IOE)^2 / 10^{18}$$

方程式 1 IOE 與 Eth 匯率交換

分潤費與分潤紅利

只要持有 IOE 的用戶，每禮拜都可以獲得提領分潤紅利，分潤紅利是分潤基金，依用戶所持有的比例所發放的，而分潤基金是所有用戶交易買賣所需要的分潤費(表格 3)所累積的。

買/賣	內含/外加	利率(%)
購買時	內含	8%
販賣時	內含	10%

表格 3 分潤費計算

開發費與使用目的

當用戶購買或是販賣時都會有一定的費用(表格 4)會支付給開發團隊，來持續開發和維持開發系統架構，開發費用的使用如(表格 5)

買/賣	內含/外加	利率(%)
購買時	外加	3%
販賣時	內含	5%

表格 4 開發費

比例	分配方案
50%	IOE 系統開發費用
10%	知名業內人士天使輪融資
20%	推廣費用
20%	營運儲備金

表格 5 費用細節

推薦制度與獎金

只要 0.02Eth 可以註冊成為推薦 VIP 的成員可以推薦用戶加入，本身可以獲得推薦的用戶分潤費的 33.33%當紅利獎金。

實作時程

代幣發行 ICO 與推廣

項目名稱	工作細項	日期
ERC20 代幣發行	開發代幣的智能合約	2018/11/15
代幣推廣	眾籌與推廣	2019/4/30
交易所上架	ICO 結束	2019/5/10

應用開發

項目名稱	工作細項	日期
第一應用上架	第一個使用 IOE 代幣的遊戲上架	2019/5/30
物品 API	物品規範與核心程式開發完成	2019/6/20
含有虛擬物品的 App	有虛擬物品交易功能的遊戲上架	2019/8/30
開放第三方應用 API	Api 對外開放使用	2019/10/1

轉為公鏈

引用的項目

1. **blockchainhub**. Decentralized Applications – dApps. blockchainhub. [線上]
<https://blockchainhub.net/decentralized-applications-dapps/>.
2. **CoinTmr**. Will the 137 billion dollar game market be the tipping point for the blockchain industry? [線上]
<https://cointmr.com/1370%E5%84%84%E7%BE%8E%E9%87%91%E7%9A%84%E9%81%8A%E6%88%B2%E5%B8%82%E5%A0%B4%EF%BC%8C%E6%9C%83%E6%98%AF%E5%8D%80%E5%A1%8A%E9%8F%88%E7%94%A2%E6%A5%AD%E7%9A%84%E5%BC%95%E7%88%86%E9%BB%9E%E5%97%8E/>.
3. **SharmaLokesh**. Facebook: An application of cloud computing.
4. A Sociability Study of Facebook Games : The Perspectives of Group Member Roles and Interpersonal Relationship Types. **Advisor: Jim Jiunde LeeD.Ph.**
5. **read01**. Game life cycle analysis: taking into account the game life cycle and the game user life cycle. [線上] <https://read01.com/zh-tw/jDKKdd.html#.XDI4llwzbIU>.
6. Study on the influence of on-line game players' persistent usage. **WuChia-Ying**.
7. Playability Impact – An Updating Strategy Analysis of World of Warcraft. **TsengYi-cheng**.
8. **zhuanlan**. Game data analysis -what can be lost due to player loss. [線上]
<https://zhuanlan.zhihu.com/p/26332219>.
9. **git**. git. [線上] <https://git-scm.com/>.
10. **docker**. Docker: Enterprise Container Platform. [線上] <https://www.docker.com/>.
11. **Linux.org**. [線上] <https://www.linux.org/>.
12. **ethereum**. Ethereum Project. [線上] <https://www.ethereum.org/>.
13. **wikipedia**. ERC-20. wikipedia. [線上] <https://en.wikipedia.org/wiki/ERC-20>.
14. **mbalib**. [線上]
<https://wiki.mbalib.com/zh-tw/%E7%A9%BA%E6%B0%94%E5%B8%81>.
15. **solidity**. [線上] <https://solidity.readthedocs.io/en/v0.5.1/#>.
16. **Ether**. etherconverter. [線上] <https://etherconverter.online/>.
17. **WhelanJoseph** 且 **MseferKamil**. ECONOMIC SUPPLY & DEMAND. 1994 年.
18. **Kning**. When new technologies meet old problems – blockchain-related financial fraud. [線上] <https://panx.asia/archives/59814>.
19. Dockerfile reference. docker. [線上]
<https://docs.docker.com/engine/reference/builder/>.
20. **github**. [線上] <https://github.com/>.
21. **bitbucket**. [線上] <https://bitbucket.org/>.
22. **wikipedia**. Smart_contract. [線上] https://en.wikipedia.org/wiki/Smart_contract.

23. jenkins. [線上] <https://jenkins.io/>.
24. Operating system. [線上] https://en.wikipedia.org/wiki/Operating_system.
25. ISO image. [線上] https://en.wikipedia.org/wiki/ISO_image.
26. SHA-2. [線上] <https://en.wikipedia.org/wiki/SHA-2>.
27. Git Repository. [線上]
<https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository>.
28. git-push. [線上] <https://git-scm.com/docs/git-push>.
29. Front and back ends. [線上]
https://en.wikipedia.org/wiki/Front_and_back_ends#Back-end_focused.
30. Google Cloud. [線上] <https://cloud.google.com/>.
31. Coding in C# in Unity for beginners. [線上]
<https://unity3d.com/learning-c-sharp-in-unity-for-beginners>.
32. Metamask. [線上] <https://metamask.io/>.
33. IOE Contract Address. [線上]
<https://etherscan.io/address/0x15eBe584D7a0c28a3a4e73E5430B08324F3296f9>.
34. IOE trade. [線上] <https://ioelink.net/?page=trade&invite=&>.
35. dApp-Quick-Start-Guide. [線上]
<https://github.com/IOElinkio/dApp-Quick-Start-Guide>.