**IOE**

**IntelligenT Operating System Exchange**

內容

[摘要 3](#_Toc535792126)

[簡介 4](#_Toc535792127)

[系統架構與節點Node 7](#_Toc535792128)

[CryptoNode 8](#_Toc535792129)

[IaasNode 9](#_Toc535792130)

[JenkinsNode 9](#_Toc535792131)

[DockerNode 9](#_Toc535792132)

[交易所Node 10](#_Toc535792133)

[CryptoNode API 與 資料 11](#_Toc535792134)

[資料結構 11](#_Toc535792135)

[用戶資料 11](#_Toc535792136)

[App資料 11](#_Toc535792137)

[系統資料 11](#_Toc535792138)

[JenkinsNode資料 12](#_Toc535792139)

[DockerNode資料 12](#_Toc535792140)

[CryptoNode API 12](#_Toc535792141)

[虛擬物品交易所 12](#_Toc535792142)

[IOE 代幣 13](#_Toc535792143)

[團隊營運資金 13](#_Toc535792144)

[去中心化程式 14](#_Toc535792145)

[程式上架 14](#_Toc535792146)

[程式改版與投票 14](#_Toc535792147)

[實作時程 15](#_Toc535792148)

[代幣發行與推廣 15](#_Toc535792149)

[應用開發 15](#_Toc535792150)

[第三方人員開發 15](#_Toc535792151)

[轉為公鏈 15](#_Toc535792152)

[引用的項目 16](#_Toc535792153)

# 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運行平台，最主要是解決虛擬物品公平性的問題．

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

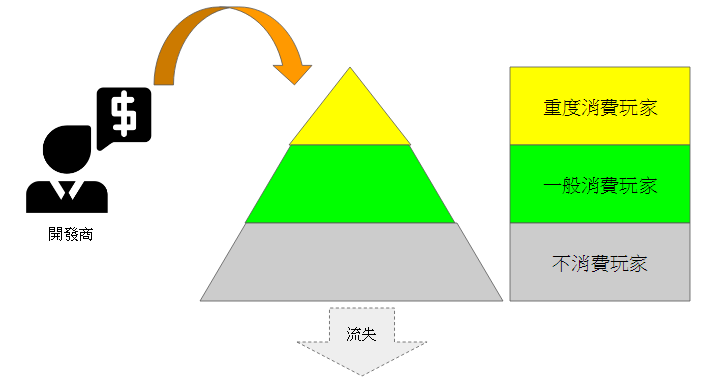
以上幾點導致虛擬物品沒有穩定的價格,導致收藏的價值低落．

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(CrtptoNode)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IaasNode 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：IaasNode 和 CryptoNode, IaasNode 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IassNode可以做操作和修改. ．

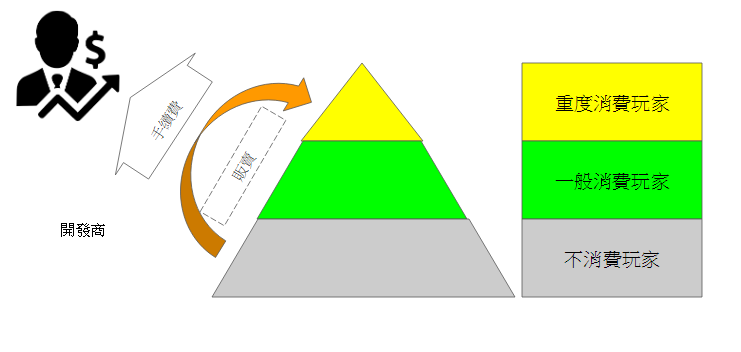
# 簡介

區塊鏈技術中的dApp,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，所以用遊戲產業來推動區塊練技術是最為可行的，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，更別說Facebook當年也是靠小遊戲捕獲大眾市場，本團隊透過區塊鏈去中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，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，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(IOE 代幣)，本代幣是透過ethereum (12)中的ERC20 (13)的規範發行的區塊練代幣，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智能和約 (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)。

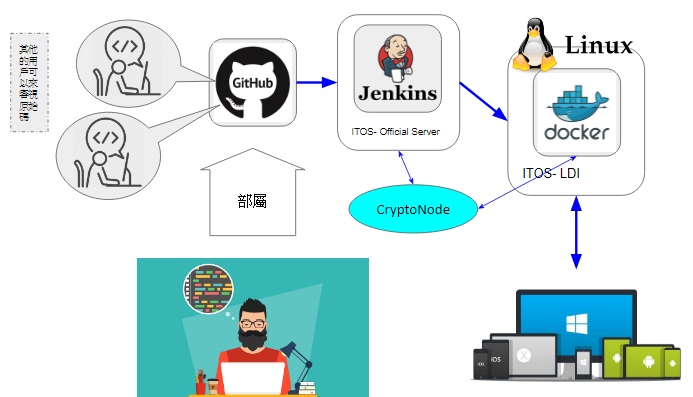
TODO我們要先找應用

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，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，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上傳到Github、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[投票機制]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[CryptoNode API]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[團隊營運資金]，而用戶玩家可以透過取得到的IOE代幣去系統內的[虛擬物品交易所]，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

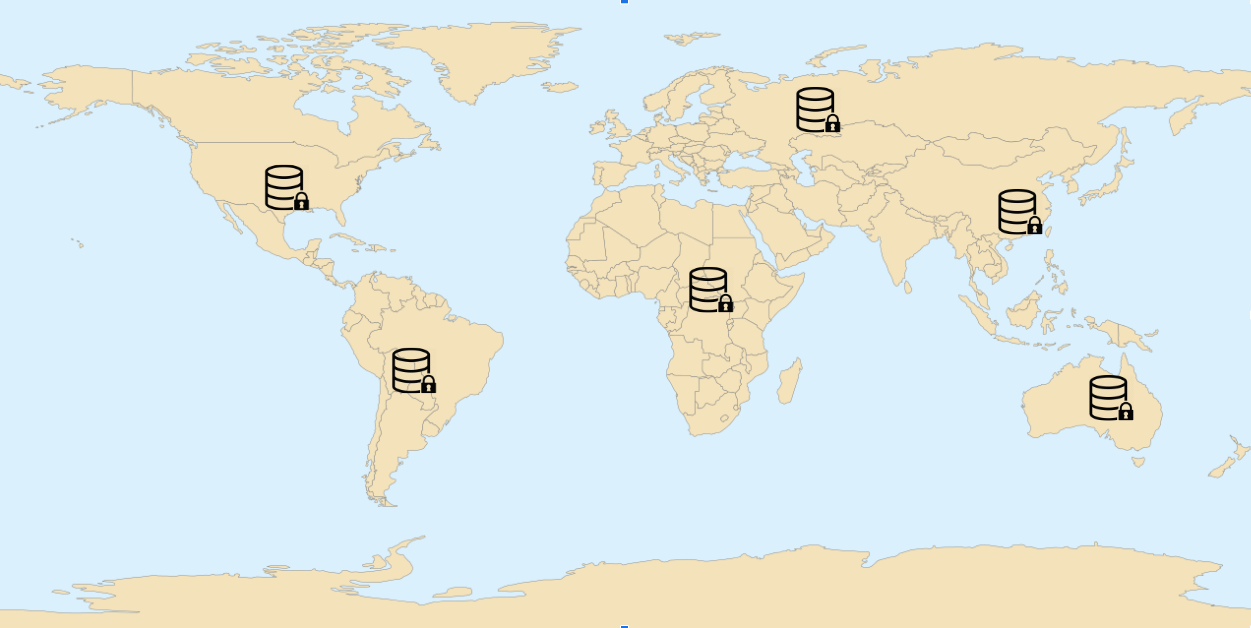
## CryptoNode

這個Node主要是紀錄本系統的[用戶資料]、[App資料]、[系統資料]、[**錯誤! 找不到參照來源。**]，要操作裡面的資料是要透過[CryptoNode API]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[CryptoNode API]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[CryptoNode API]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，現在在亞洲使用的話，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會發生的。

CryptoNode裡的紀錄[JenkinsNode資料**錯誤! 找不到參照來源。**]，這是紀錄IOE團隊開發的Jenkins伺服器,當然這個伺服器運行在[IaasNode]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

## IaasNode

IaasNode是一種Linux作業系統的運行環境，它是由IOE團隊釋出的Linux Image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[CryptoNode API]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，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

### DockerNode

#### 投票機制

### 交易所Node

# CryptoNode API 與 資料

## 資料結構

### 用戶資料

|  |
| --- |
| {  "accpuntid1": {  "apps": {  "appId1": {  "APP資料": "xxxx"  },  "@IRC30s":{  "irc30Id":11  }  }  }  } |

### App資料

### 系統資料

### JenkinsNode資料

### DockerNode資料

## CryptoNode API

## 虛擬物品交易所

# IOE 代幣

## 團隊營運資金

# 去中心化程式

## 程式上架

## 程式改版與投票

# 實作時程

## 代幣發行與推廣

## 應用開發

## 第三方人員開發

### 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#.XDl4llwzbIU.

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.