Kang Da

康 大 資 訊

傳統 Web-Based 應用程式往原生雲應用程式之移轉
以 Cloud Foundry 實作分享

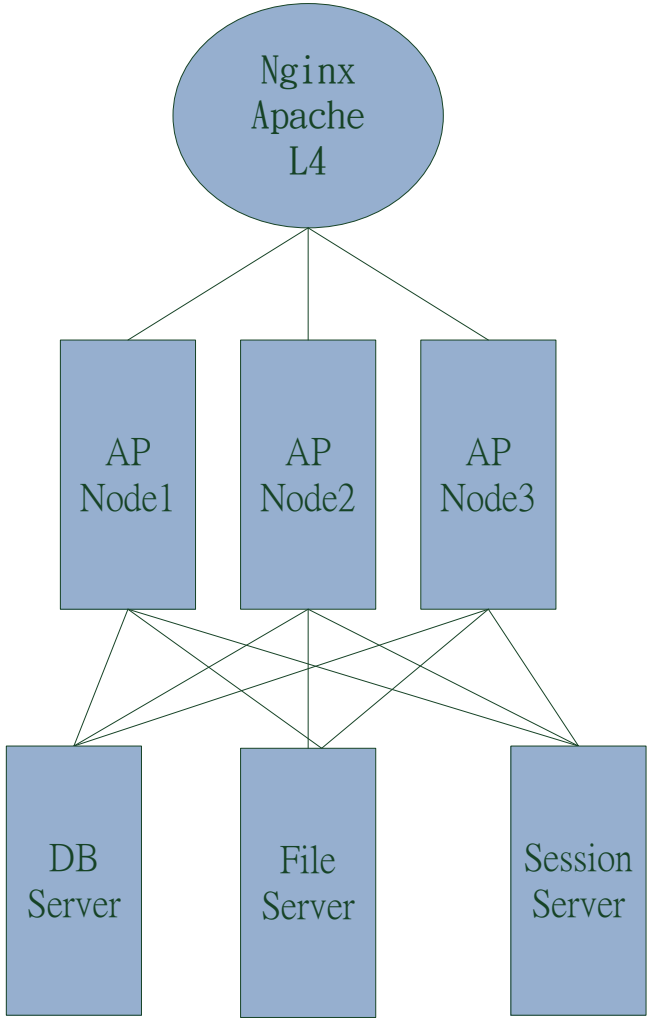
Jul 12, 2016

張宇華

大綱

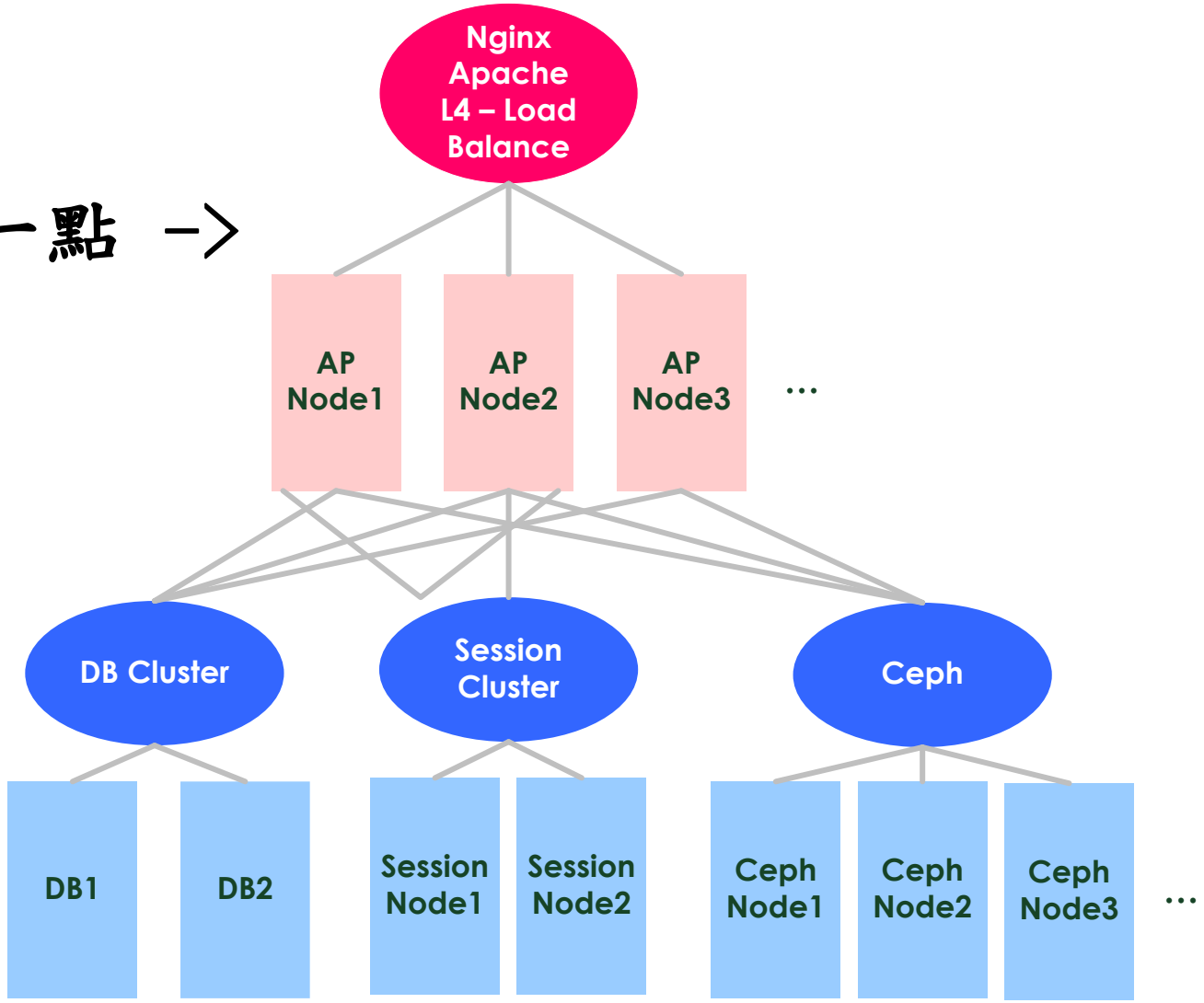
- 傳統Web-Based應用程式架構
- NIST雲端運算特徵
- The Twelve-Factors
- Docker
- The Cloud Foundry Architecture
- HPE Helion Stackato Overview
- 實作及Demo

傳統Web-Based應用程式架構



傳統Web-Based應用程式架構

更進階一點 ->



傳統Web-Based應用程式架構

- 前面不管是複雜或是簡單的應用架構，統統得由專業的架構師／開發設計人員來處理
- 資源的調度，很不彈性
- 無法確保服務水準

傳統Web-Based應用程式架構

- 為了解決問題
  - Cloud Foundry
    - HPE Helion Stackato
  - Rancher
  - DC/OS
  - Docker Swarm / SwarmKit
  - Kubernetes
  - ….

NIST雲端運算特徵

- 隨需應變自助服務(On-demand Self-service)
  - 消費者可依據使用需求狀況自行使用雲端服務，不需再透過雲端供應者與之互動。
- 網路使用無所不在(Broad Network Access)
  - 網路使用無所不在，亦即雲端供應者服務可隨時在網路取用，且使用者端無論大小，均可透過標準機制使用網路。

NIST雲端運算特徵 Cont.

- 共享資源池(Resource Pooling)
  - 資源彙整讓雲端供應者透過多租戶模式(Multi-tenancy)服務消費者，依據消費者要求，來指派或重新指派實體及虛擬資源，在所在地獨立性的概念下，消費者通常不知道所有資源確切位置，只可能掌握國家、州或資料中心等大範圍區域地點。其中資源包括儲存、處理、記憶、網路頻寬和虛擬機等。

NIST雲端運算特徵 Cont.

- 快速重新佈署靈活度(Rapid Elasticity)
  - 彈性亦即能因應需求彈性且快速調整資源規模大小，對消費者而言，所提供的這種能力似乎是無限的，可以在任何時間被購買任何數量。
- 服務可計算(Measured Service)
  - 計算服務量測中，雲端服務各層次均由雲端供應者掌控與監管，這對於計費、存取控制、資源優化、處理能力規畫及其他工作相當重要，確保資源使用可被監測、被控制和被報告，為供應者和消費者雙方提供透明化服務使用資訊。

12 Factors

- 1. Codebase
  - One codebase tracked in revision control, many deploys
- 2. Dependencies
  - Explicitly declare and isolate dependencies
- 3. Config
  - Store config in the environment
- 4. Backing Services
  - Treat backing services as attached resources

12 Factors Cont.

- 5. Build, release, run
  - Strictly separate build and run stages
- 6. Processes
  - Execute the app as one or more stateless processes
- 7. Port binding
  - Export services via port binding
- 8. Concurrency
  - Scale out via the process model

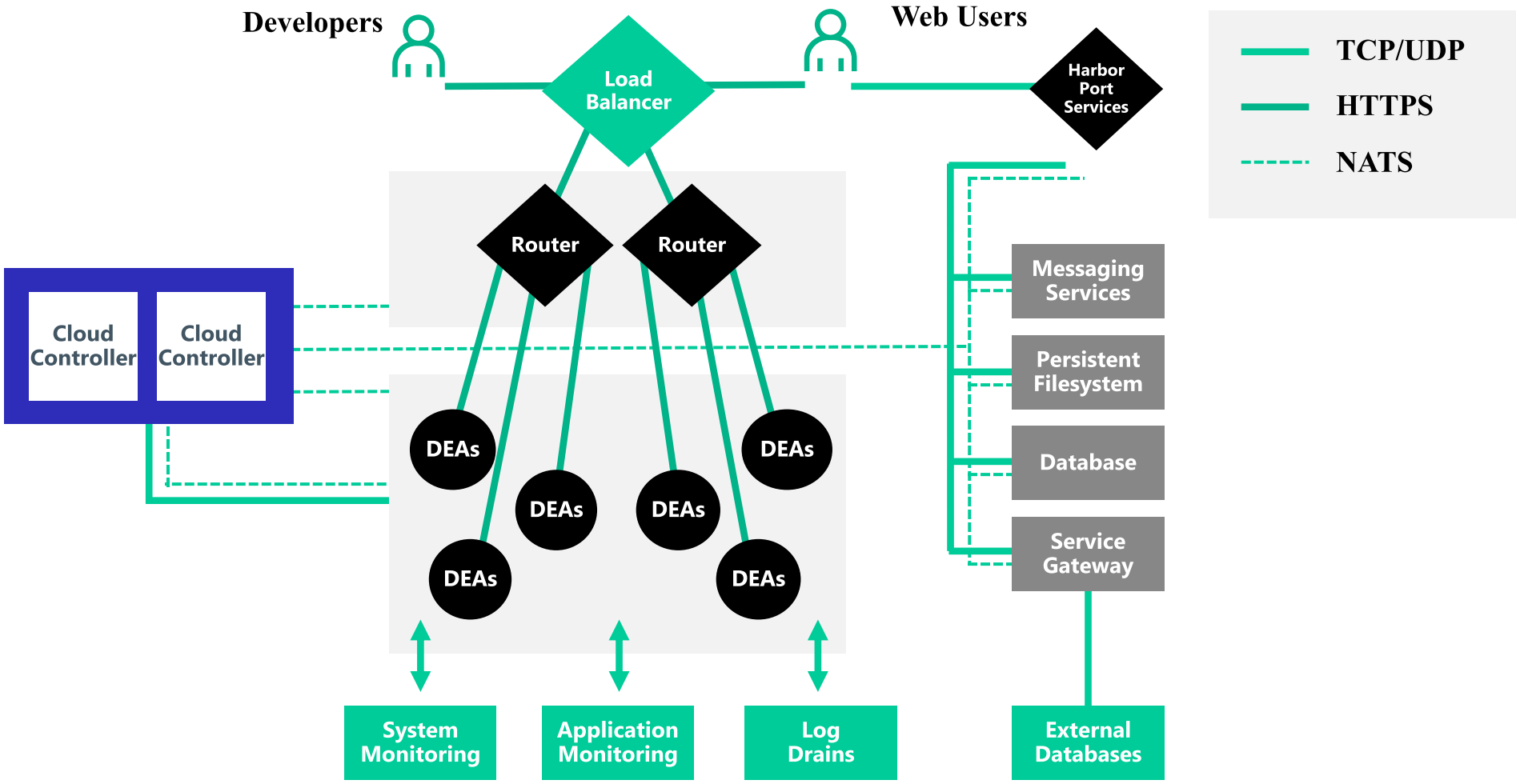
12 Factors Cont.

- 9. Disposability
  - Maximize robustness with fast startup and graceful shutdown
- 10. Dev/prod parity
  - Keep development, staging, and production as similar as possible
- 11. Logs
  - Treat logs as event streams
- 12. Admin processes
  - Run admin/management tasks as one-off processes

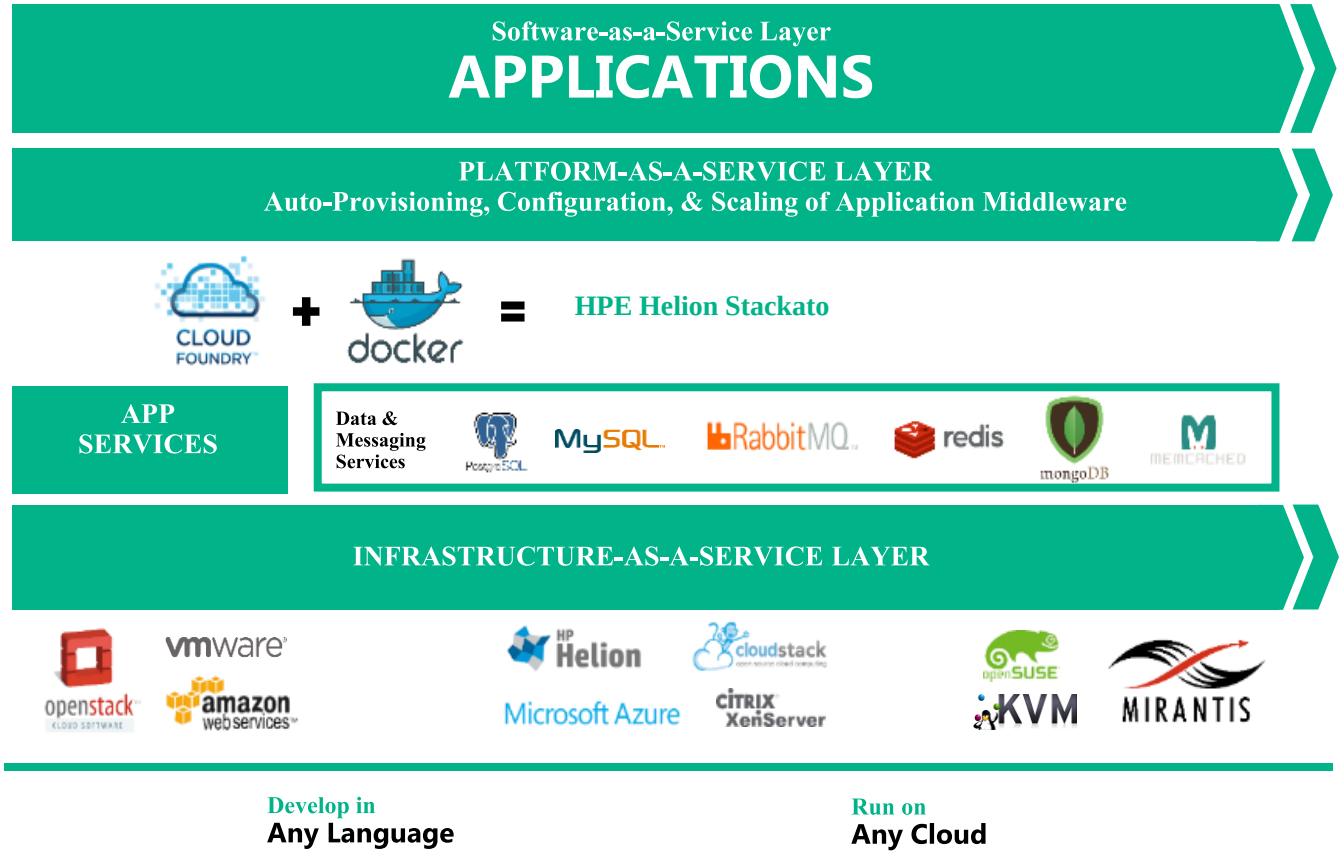
Docker

- Docker透過Container技術可以將任何小程式打包成可獨立執行的映像檔，發布到任何可執行Docker的平臺上執行。
- 降低原生雲應用程式的開發難度

The Cloud Foundry Architecture



HPE Helion Stackato Overview



- **Faster time to market**
- **Productivity and innovation**
- **Cost reduction**

A PaaS for IT & Developers

- **Developers don't manage plumbing**
- **Enables self-service deployment**
- **Automatic scaling**

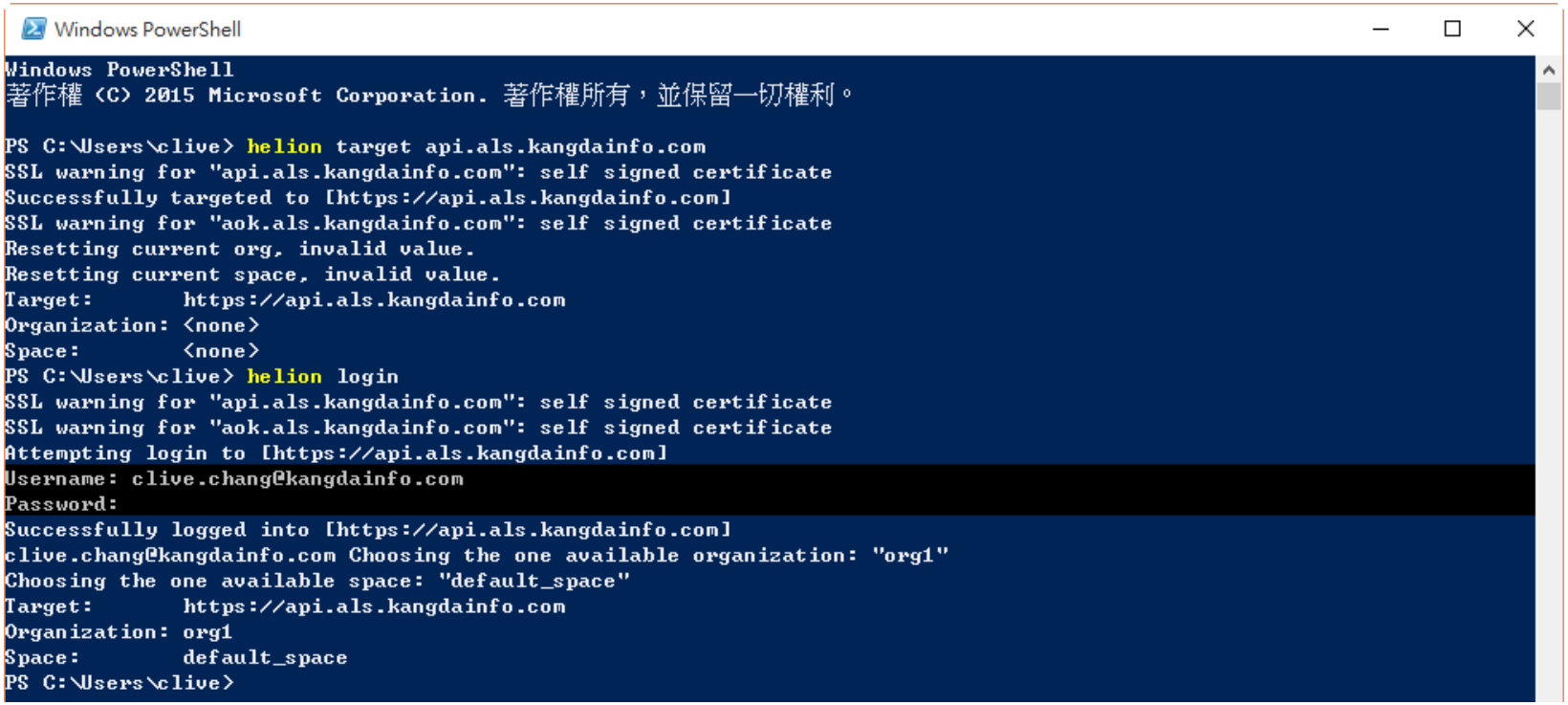
Basic compute, storage, networking, VM automation only

實作及Demo

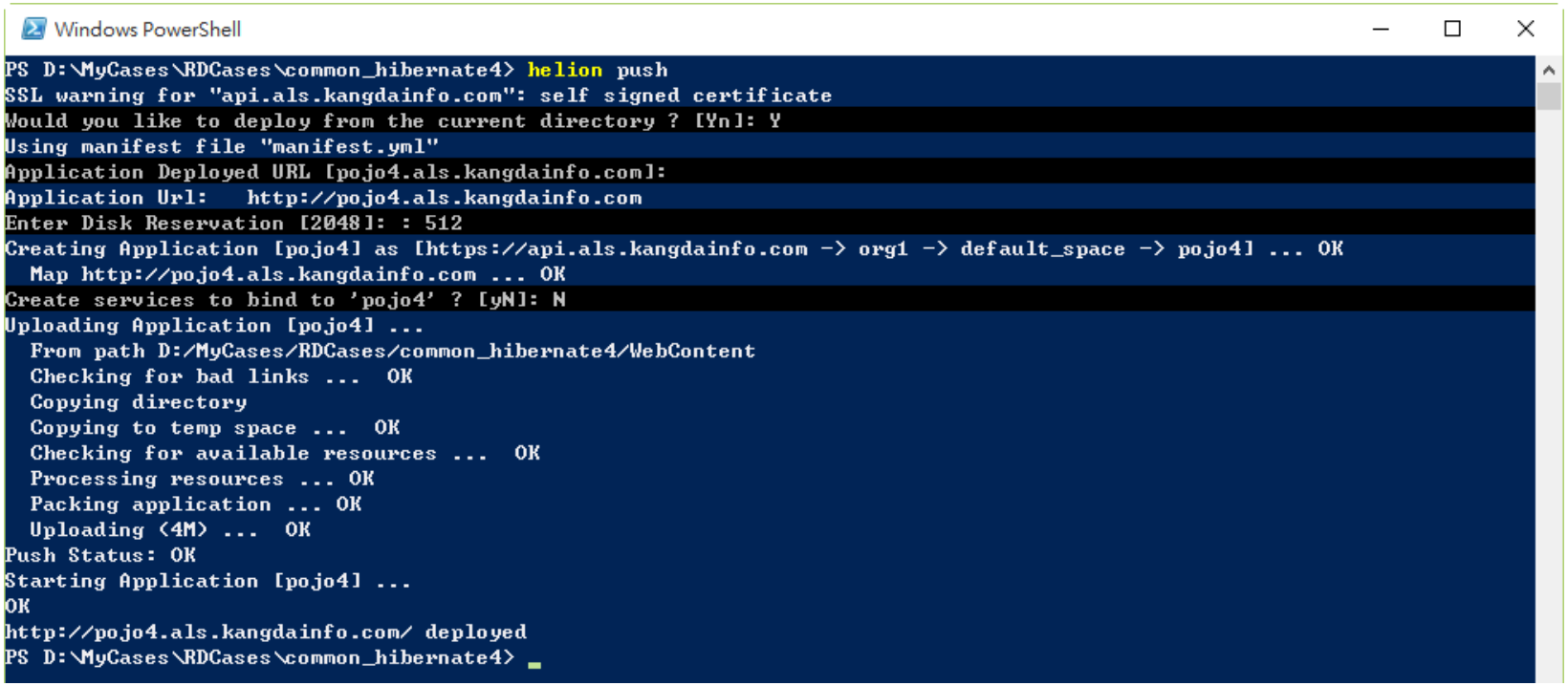
Practice :

- Step 0: Create Application Custer
- Step 1: Download and Install the CLI
- Step 2: Logon the Cluster with CLI
- Step 3: Push your app
- Build Tool Integration
  - Eclipse Cloud Foundry Plugin
  - Visual Studio Cloud Foundry Plugin

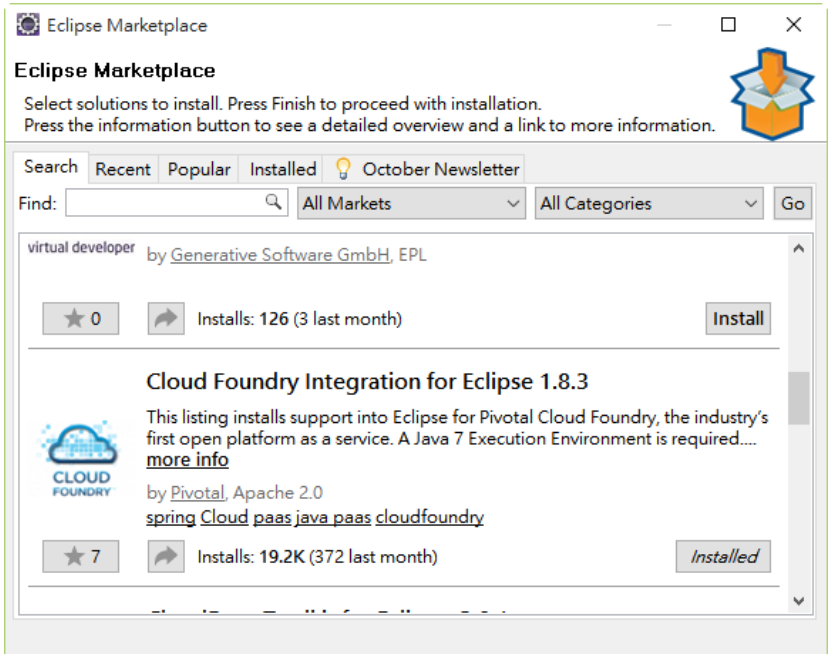
Logon the Cluster with CLI

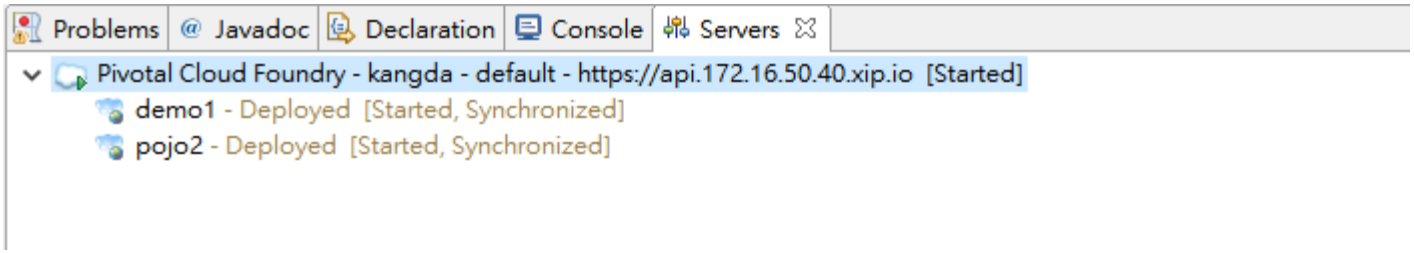


Push Application

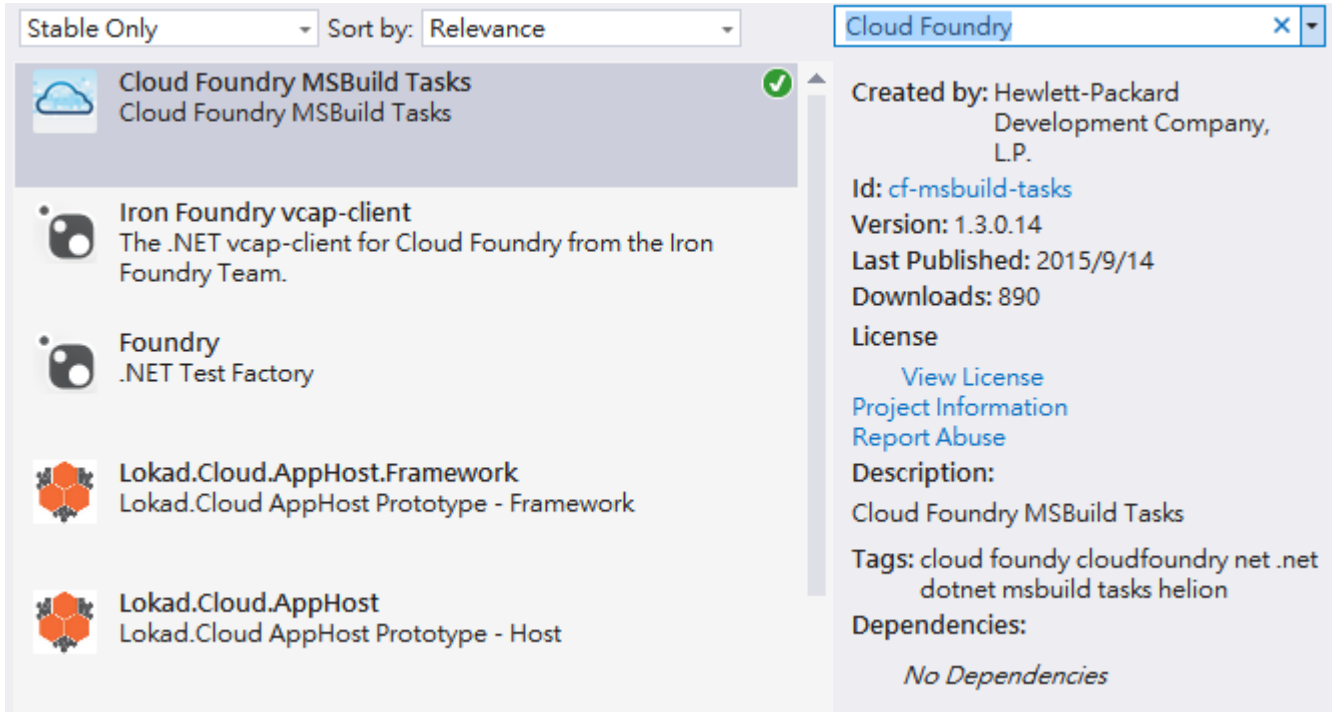


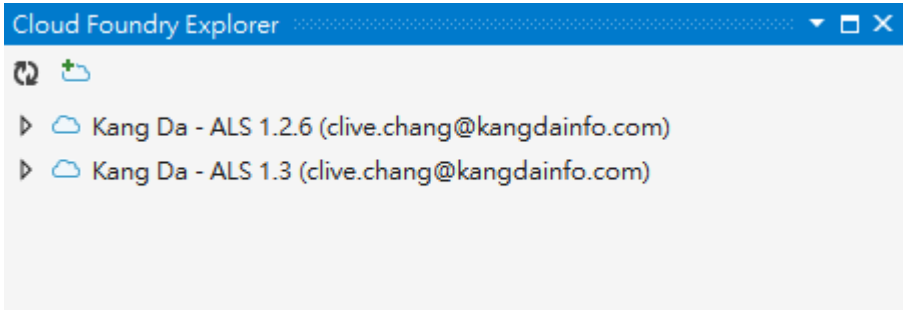
Eclipse Plugin





Visual Studio Plugin





References

- **OpenStack.org**
  - http://docs.openstack.org
- **HP Helion Documents**
  - http://docs.hpcloud.com
- **The Twelve-Factor App**
  - http://12factor.net
- **Pivotal Cloud Foundry**
  - http://docs.cloudfoundry.org
- **iThome**
  - http://www.ithome.com.tw/news/95752
- 雲端開發測試平台
  - https://www.cloudopenlab.org.tw