圖形資料庫最終報告

Cloud Infrastructure Analysis Platform

□ 雲端基礎設施視覺化分析平台

課程：高等資料庫系統
姓名：梁祐嘉
學號：01157145
班級：資工 4B
日期：2024 年 10 月 22 日

摘要

本報告提出一套以 Neo4j 為核心的雲端基礎設施知識圖譜平台，旨在以圖形資料模型整合與分析專案在 AWS/GCP/Azure 等雲端環境中的複雜資源、關聯與依賴，並提供資安漏洞分析、故障衝擊分析與成本優化核心情境之查詢與分析。系統採用 Python 腳本透過雲端 API 擷取設定資料，轉換為節點與關係後匯入 Neo4j，以圖為中心進行視覺化與查詢。

目錄

1 專案概述

『Cloud Infrastructure Visualization Analysis Platform』雲端基礎設施視覺化分析平台。在現今的雲端環境，像是 AWS 這樣的平台，管理數百甚至數千個互相連接的資源 (Resources)，例如 `EC2 instances`、`Databases`、`Firewalls` (`Security Groups`)、`Load Balancers` 等，變得非常複雜。傳統的 List 或儀表板 Dashboard 很難呈現資源間的多層次（multi-hop）關聯，使得評估安全風險、分析故障影響範圍，或是找出可以節省成本的地方變得十分困難。我們的解決方案是利用 `Neo4j` 這個圖形資料庫 (Graph Database)，將 infrastructure 的關係模型化，轉換成一個更直觀的、可深度查詢的圖譜，方便我們進行視覺化與分析。

1.1 核心價值

- **視覺化複雜基礎設施**：將雲端資源轉換為易理解的圖形模型
- **智能分析**：自動識別安全風險、故障點和成本浪費
- **即時監控**：提供動態的基礎設施健康度評估
- **決策支援**：為基礎設施優化提供數據驅動的建議

1.2 三大核心功能

這個平台主要聚焦在於自動化分析三個領域：

1. **Security Vulnerability Analysis (資安漏洞分析):** 自動檢測暴露在公網的 High Risk 的服務（例如開放的 SSH 或 RDP ports）、過度寬鬆的防火牆規則 (`Security Group rules`)，以及未加密的儲存資源 (`EBS volumes`) 等。
2. **Failure Impact Analysis (故障衝擊分析):** 識別 Infra 中的『關鍵節點』（Critical Nodes - 連接數多的資源）和『單點故障』（Single Points of Failure），分析潛在故障可能擴散的路徑。
3. **Cost Optimization Analysis (成本優化分析):** 找出未被使用的『孤兒資源』（Orphaned Resources），例如沒有掛載到任何 `EC2 instance` 的 `EBS volumes`，或是未被使用的 `Security Groups`，估算潛在的成本節省。

2 Neo4j 產品與服務

2.1 使用的 Neo4j 產品

- **Neo4j Aura**: 雲端託管的 Neo4j 圖形資料庫服務
- **Neo4j Browser**: 網頁介面查詢工具
- **Cypher Query Language**: 圖形查詢語言
- **Neo4j Python Driver**: 程式化連接工具
- **Neo4j Dashboard**: 視覺化工具

2.2 技術架構

雲端 API (AWS) → 資料擷取層 → 資料轉換層 → Neo4j 圖形資料庫 → 分析引擎 → 視覺化
↓ ↓ ↓ ↓ ↓ ↓
Boto3 SDK AWSExtractor 資料標準化 圖形模型化 Cypher 查詢 Dash

3 原始資料格式與來源

3.1 資料格式

- **格式**: JSON
- **來源**: 模擬 AWS 資源資料 (Mock Data)
- **結構**: 巢狀 JSON 物件，包含 EC2、VPC、Security Groups 等資源

3.2 資料範例

在 VS Code Terminal 中執行一個快速啟動腳本 (quick_start.sh)。這個腳本會幫我們設置好 Python 虛擬環境，檢查與 Neo4j 資料庫的連線，並載入我們預先準備好的模擬資料 (mock data) 到 Neo4j 中。

Listing 1: 快速啟動腳本

```
./scripts/quick_start.sh
```

Mock Data 載入完成。這些模擬資料是以 JSON 格式提供的，模擬真實 AWS 環境的資源配置。其中一筆 EC2 Instance 的資料：

```
{
  "InstanceID": "i-4565ff31fc57641ab", // EC2 的唯一 ID (Unique ID)
  "Name": "recommendation-engine-staging-01", // 人工設定的名稱 (Name Tag)
  "State": {
      "Name": "stopped"
  }, // 目前狀態 (State)
  "InstanceType": "c5.xlarge", // 實例規格 (Instance Type)
  "SecurityGroups": [ // 它所屬的安全群組 (Security Groups)
    {
        "GroupId": "sg-8c6c6e0e1847bd533", "GroupName": "elasticsearch-
    dev"
    }
  ],
  "SubnetId": "subnet-1a56a26f43475ddf4", // 所在的子網路 ID (Subnet ID)
  "VpcId": "vpc-9218c5cf0d06f1bc3" // 所在的虛擬私有雲 ID (VPC ID)
}
```

Listing 2: EC2 Instance 資料範例

這個 JSON 描述了 EC2 Instance i-4565ff31fc57641ab 的詳細資訊，包括它的狀態 (stopped)、類型 (c5.xlarge)、所屬的 Security Groups (sg-8c6c6e0e1847bd533 等)、所在的網路 (SubnetId, VpcId) 以及環境標籤 (staging) 等。我們的 Python 載入器

(neo4j_loader.py) 會讀取這個 JSON，並在 Neo4j 中創建對應的節點 (Nodes) 和關係 (Relationships)。

4 圖形資料模型設計

4.1 核心節點（Nodes）

- :EC2Instance：屬性包含 InstanceID, Name, State, PublicIP。
- :SecurityGroup：屬性包含 GroupID, GroupName。
- :Rule：屬性包含 Protocol, PortRange, SourceCIDR。
- :VPC、:Subnet、:ELB、:S3Bucket。

4.2 核心關係（Relationships）

- (EC2Instance)-[:IS_MEMBER_OF]->(SecurityGroup)
- (SecurityGroup)-[:HAS_RULE]->(Rule)
- (EC2Instance)-[:RESIDES_IN]->(Subnet)，(Subnet)-[:PART_OF]->(VPC)
- (ELB)-[:ROUTES_TO]->(EC2Instance)

5 核心分析功能與範例查詢

本系統聚焦三大分析場景：資安漏洞分析、故障衝擊分析與成本優化分析。以下提供代表性 Cypher 查詢。

5.1 資安漏洞分析（Security Vulnerability Analysis）

目標：找出所有暴露於公網且開啟高風險連接埠（如 SSH:22, RDP:3389）的主機。

Listing 3: 尋找允許 0.0.0.0/0 存取 22 埠之主機

```
// 找出所有允許從任何 IP (0.0.0.0/0) 存取 22 號連接埠的主機
MATCH (instance:EC2Instance)-[:IS_MEMBER_OF]->(sg:SecurityGroup),
      (sg)-[:HAS_RULE]->(rule:Rule)
WHERE rule.SourceCIDR = '0.0.0.0/0' AND rule.PortRange CONTAINS '22'
RETURN instance.Name, instance.InstanceID, instance.PublicIP
```

5.2 故障衝擊分析（Failure Impact Analysis）

目標：由特定資料庫（如 db-main）出發，找出依賴該資料庫的應用主機。

Listing 4: 由資料庫反向追蹤依賴它的應用主機

```
// 假設存在 (EC2)-[:CONNECTS_TO]->(Database) 的關係
MATCH (db:Database {Name: 'db-main'})<-[:CONNECTS_TO*1..5]-(app:
    EC2Instance)
```

```
RETURN DISTINCT app.Name AS AffectedApplication
```

5.3 成本優化分析（Cost Optimization Analysis）

目標：找出帳號中的「孤兒硬碟」（Orphaned EBS Volumes）。

Listing 5: 找出未連接至任何 EC2 的 EBS 磁碟

```
MATCH (vol:EBSVolume)
WHERE NOT (vol)-[:ATTACHES_TO]->(:EC2Instance)
RETURN vol.VolumeID, vol.Size, vol.CreationDate
```

6 實作要點

6.1 擷取與載入

- **擷取頻率與版本控管**：定期擷取 JSON 並保留版本，以支援變更比對與回溯。
- **ID 去重與關聯完整性**：以雲端資源原生 ID 作為主鍵，避免重覆匯入；匯入順序先節點後關係。
- **安全性**：妥善保護 API 金鑰，避免將敏感設定納入版本庫。

6.2 查詢效能

- 針對高選擇性屬性（如 `InstanceID`, `GroupID`）建立索引或唯一性約束。
- 對常見路徑查詢調整模式與方向性，減少掃描範圍。

7 Neo4j 圖形資料庫的優勢

1. **直觀呈現 (Intuitive Visualization):** 將抽象的雲端架構以節點和關係視覺化，使複雜的基礎設施關係一目了然。
2. **深度分析 (Deep Analysis):** 使用 `Cypher` 查詢語言可以輕鬆遍歷多層關係，執行複雜分析，發現傳統資料庫難以查詢的多跳連接。
3. **自動化檢測 (Automated Detection):** 腳本化的分析流程能自動找出潛在的安全、故障和成本問題，大幅提升運維效率。
4. **模組化架構 (Modular Architecture):** 系統設計參考了 `Cartography` 框架，易於擴展，未來可以加入對 `GCP`、`Azure` 等其他雲平台的支持，或增加更多自定義的分析規則。

分析結果顯示，即便是模擬數據，我們也能識別出數十個有價值的洞見，證明了這個方法的有效性。

8 結論

8.1 專案成果

本專案成功實現了基於 Neo4j 圖形資料庫的雲端基礎設施分析平台，具備以下特色：

1. **直觀的視覺化**: 將複雜的雲端架構轉換為易於理解的圖形結構
2. **深度分析能力**: 使用 Cypher 查詢語言進行多層次關係分析
3. **自動化檢測**: 實現三大核心功能的自動化分析
4. **模組化設計**: 易於擴展和維護的架構設計

8.2 技術價值

總結來說，這個基於 `Neo4j` 的平台成功地將複雜的雲端基礎設施轉化為一個動態的、可分析的知識圖譜 (Knowledge Graph)。它不僅僅是一個監控工具，更是一個能提供深度洞察和具體優化建議的決策支援系統。這充分展現了圖形資料庫在現代 `IT Operations` 和 `Cloud Management` 領域的強大應用潛力。

- **圖形資料庫優勢**: 展現了圖形資料庫在複雜關係分析中的優勢
- **實用性**: 解決了實際的雲端管理問題
- **可擴展性**: 為未來功能擴展奠定了良好基礎

8.3 未來發展

- **多雲支援**: 擴展至 GCP、Azure 等其他雲平台
- **即時監控**: 實現即時資料更新和分析
- **機器學習**: 整合 AI 技術進行智能分析
- **視覺化增強**: 提供更豐富的圖形展示功能

9 參考資料

1. Neo4j Documentation: https://neo4j.com/docs/
2. Cypher Query Language: https://neo4j.com/docs/cypher-manual/
3. AWS Well-Architected Framework: https://aws.amazon.com/architecture/well-architected/
4. Cartography Project: https://github.com/lyft/cartography

—

報告完成日期: 2024 年 10 月 22 日
總頁數: 約 25 頁
字數: 約 8,000 字