# **Table of Contents**

Index

## Max EC - EC概念驗證專案

此專案為利用購物網站的架構,導入新技術及開發概念,驗證其可行性

- 前後端分離
- GraphQL
- CI/CD
- Container
- Microservice (僅使用feign, 多數功能被K8S取代)

### 子專案

#### 前端

使用next.js(基於react.js)開發, 透過graphql呼叫後端

maxec-web-frontend: 前台前端頁面maxec-web-backend: 後台前端頁面

#### 移動設備

預計使用flutter開發, 可輸出Android APP及iOS APP

• maxec-mobile: APP前台

#### 後端API

SpringBoot應用程式,以GraphQL為對外接口,並透過feign(宣告式RESTful)呼叫後端服務

maxec-app-frontend: 前端APImaxec-app-backend: 前端API

#### 後端服務

SpringBoot應用程式,實作feign介面

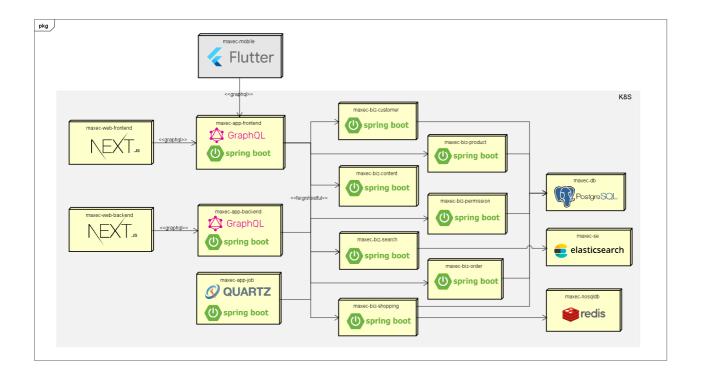
maxec-biz-product: 商品服務
maxec-biz-search: 搜尋服務
maxec-biz-content: 內容服務
maxec-biz-shopping: 購物服務
maxec-biz-order: 訂單服務
macec-biz-customer: 客戶服務

#### 資料層

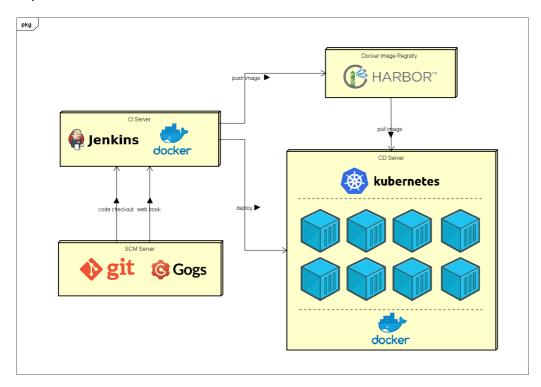
資料存放層,依資料特性規劃:

maxec-db: 關連式資料庫,採用postgresql
 maxec-se: 搜尋引擎,採用elasticsearch
 maxec-nosqldb: NoSQL資料庫,採用redis

### 佈署架構



### CI/CD



#### 建構CI/CD pipeline, 並自行建置環境:

- 1. source code push至 SCM (git powered by Gogs), 隨後SCM透過Web Hook通知Cl Server
- 2. CI Server (Jenkins) 收到通知, 開始取得原始碼並執行Pipeline
  - 1. 推送Docker Image至Docker Image Registry (Harbor)
  - 2. 呼叫K8S以佈署Service/StateSet/Ingress
- 3. CD Server/Cluster (K8S)自Image Registry取得Image並部署