

វិទ្យា Microservices

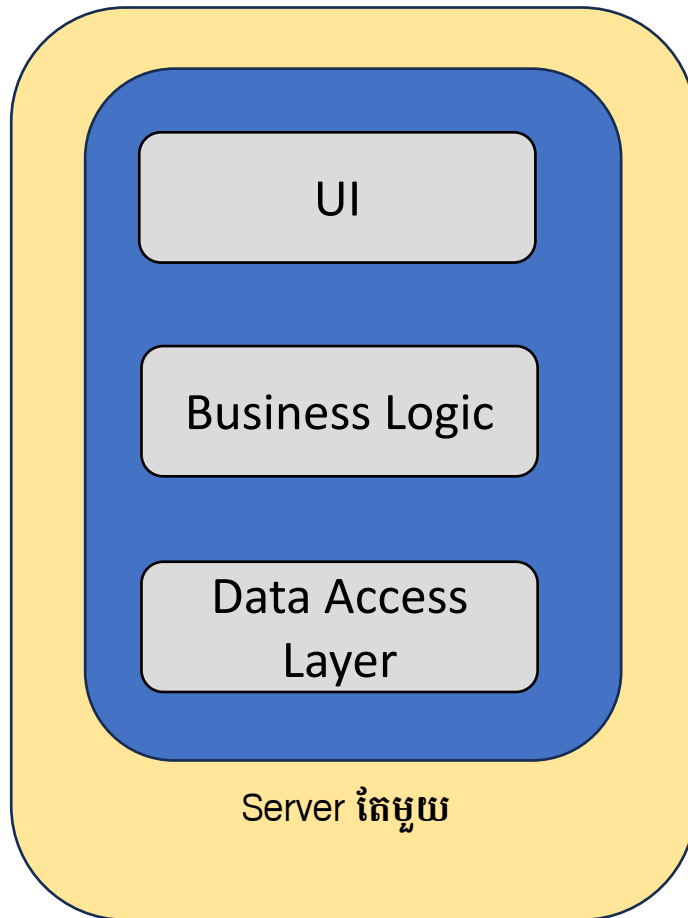
Advance Java Developer

Spring Boot
Spring Cloud
Docker
Kubernetes
OAuth2
MongoDB and PostgreSQL

៦ កក្កដា ២០២៣



Monolithic

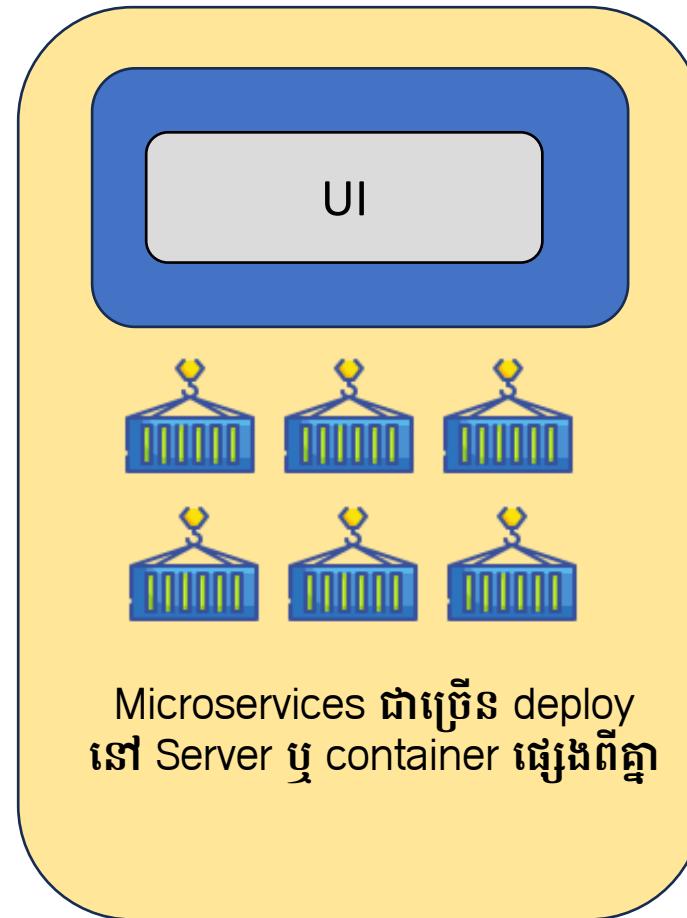


Server តែមួយ



DB

Microservices

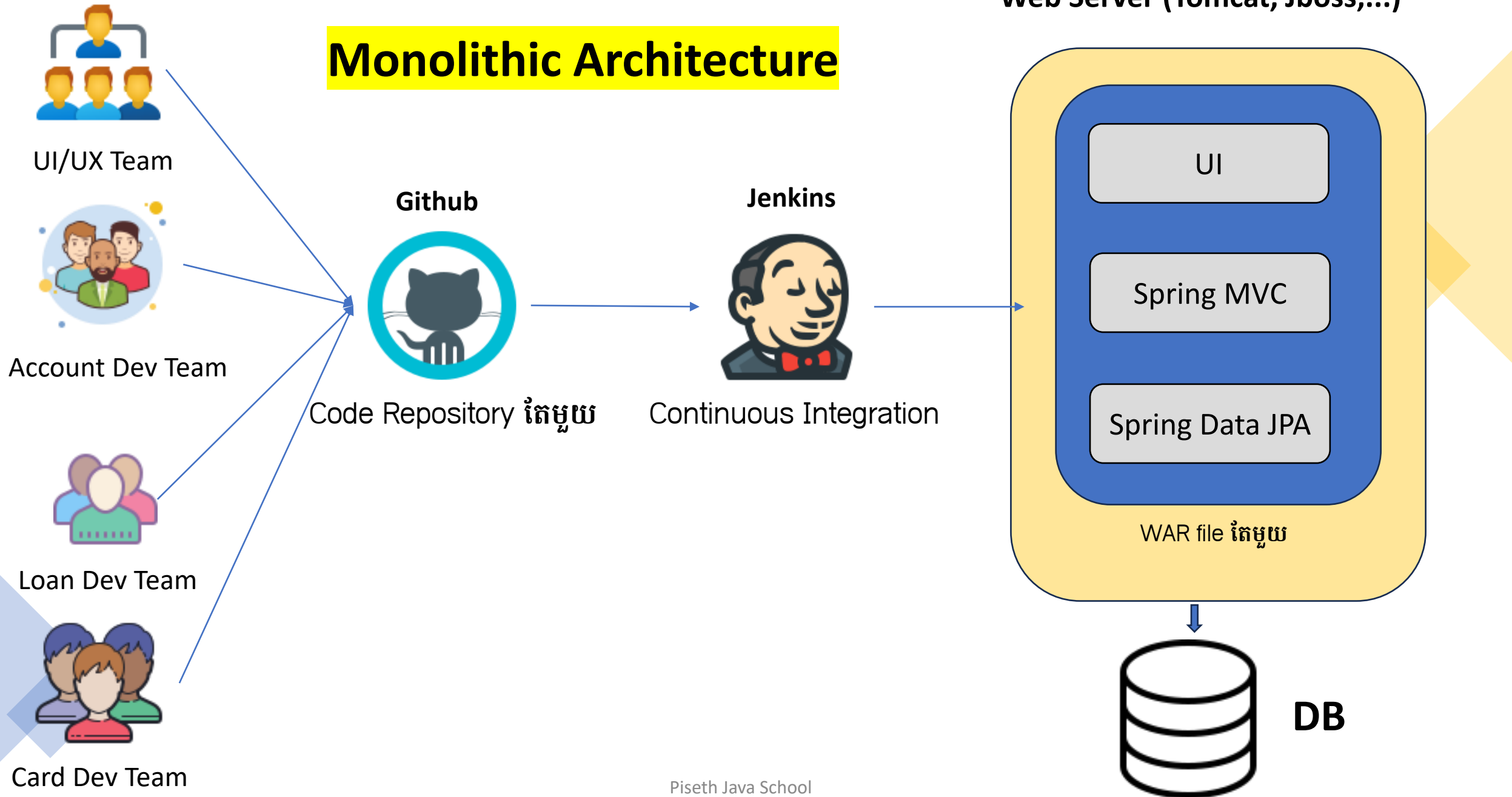


Microservices ជាច្រើន deploy
នៅ Server ឬ container ផ្សេងពីគ្នា



DBs

Monolithic Architecture



Monolithic Architecture

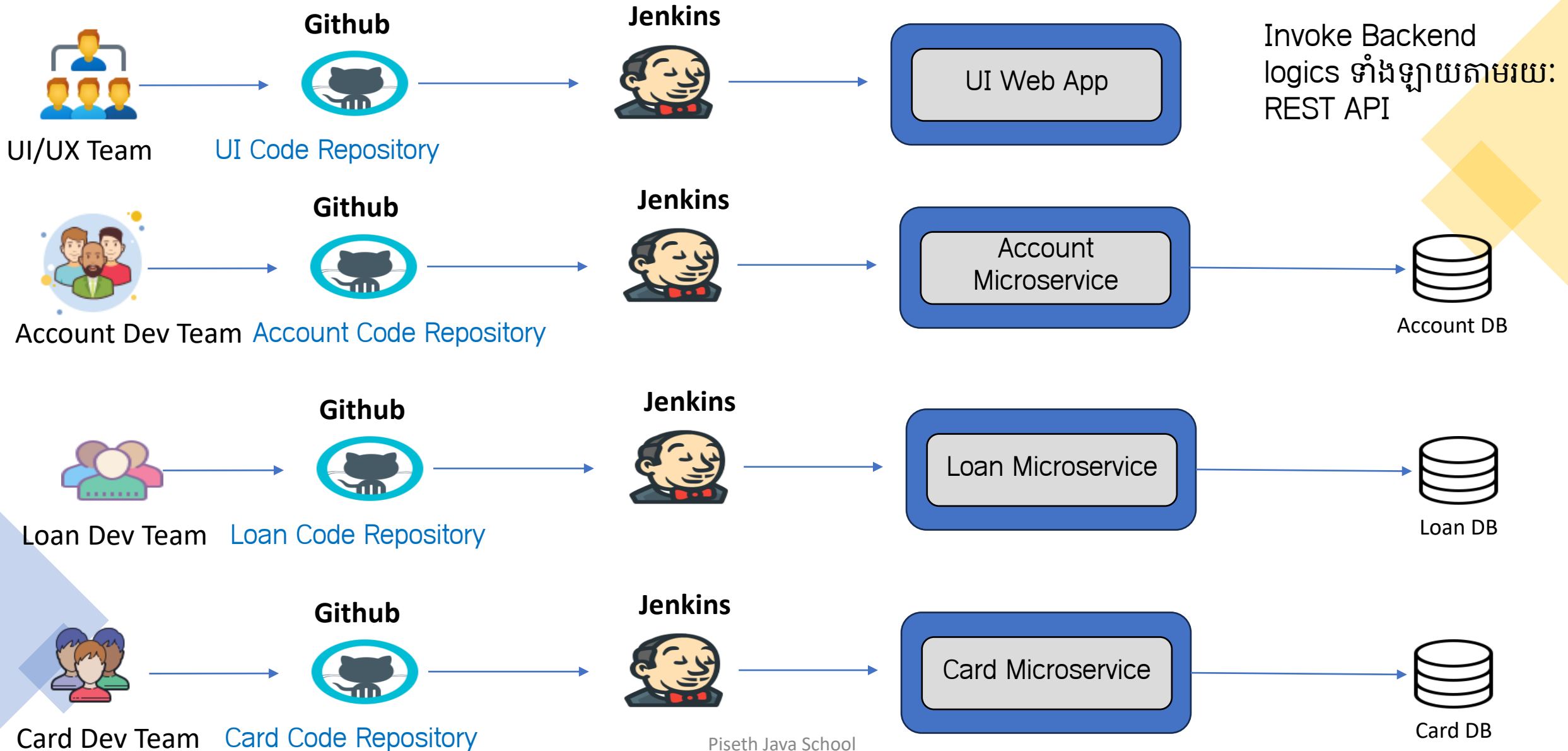
គុណសម្បត្តិ៖

- ងាយស្រួល Develop និង Deploy សម្រាប់ Application តូចៗ
- មាន Cross Cutting Concern តិចតួច (Log , Security, Configuration,...)
- ការពន្យាពេលឆ្លើយតបដោយសារ Network Communication មានតិចតួច

គុណវិបត្តិ៖

- ពិបាកក្នុងការតាមអោយទាន់ Technology ថ្មីៗ
- ការអនុវត្តតាម Agile Methodology នៅមានកម្រិត
- ប្រើ Single Code Base ហើយពិបាកក្នុងការថែទាំ (Maintain)
- ការ Update ឬ ថែម feature តូចមួយក៏ទាមទារអោយ Deploy System ទាំងស្រុងឡើងវិញ
- គ្មាន Fault Tolerance (System មិនអាចបន្តដំណើរការដោយគ្មានការរំខាននៅពេលផ្នែកណាមួយមានបញ្ហា)

Microservices Architecture



Microservices Architecture

គុណសម្បត្តិ៖

- ងាយស្រួល Develop, Test និង Deploy Service នីមួយៗ
- ការអនុវត្តតាម Agile Methodology មានប្រសិទ្ធភាពខ្ពស់
- ងាយស្រួលក្នុងការ Scale Application (Horizontal Scale)
- Team ទាំងឡាយអាច Develop ដំណោះស្រាយបាន
- Service នីមួយៗអាចជ្រើសរើស Language ឬ Technology ផ្សេងគ្នាបាន។ល។

គុណវិបត្តិ៖

- ភាពស្មុគស្មាញនៃ Architecture
- Infrastructure ច្រើន
- សុវត្ថិភាព (Security Concern)

តើអ្វីជា Microservices?

Microservices ជាវិធីសាស្ត្រនៃការ develop application ទៅជា Service មួយដែលតូចល្អម អាច run ក្នុង process ខ្លួនឯង ហើយការធ្វើទំនាក់ទំនងទៅកាន់ Service ដទៃតាមបែប lightweight mechanisms ការ develop គឺតែត្រឹមផ្នែកណាមួយនៃ Business Logic ហើយ Deploy ដោយដាច់តែឯងដោយប្រព័ន្ធស្វ័យប្រវត្តិពេញលេញ។

ប្រែសម្រួលពីអត្ថបទរបស់លោក James Lewis and Martin Fowler's

ហេតុអ្វី Spring ជា Framework ល្អបំផុតដើម្បី build Microservices?

Spring is the most popular development framework for building java-based web applications & services. From the Day1, Spring is working on building our code based on principles like loose coupling by using dependency injection. Over the years, Spring framework is evolving by staying relevant in the market.

- Building small services using SpringBoot is super easy & fast
- Spring Cloud provides tools for dev to quickly build some of the common patterns in Microservices
- Provides production ready features like metrics, security, embedded servers
- Spring Cloud makes deployment of microservices to cloud easy
- There is a large community of Spring developers who can help & adapt easily

តើ SPRING BOOT ជាអ្វី?

ការប្រើ SPRING BOOT ដើម្បី develop Microservices

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

STAND ALONE SPRING APPS:

Creating Standalone Spring applications/REST services is super quick & easy

NO NEED TO DEPLOY INTO A SERVER:

Embed Tomcat, Jetty or Undertow servers available and the deployment happens directly

STARTER PROJECTS:

Starters projects are a set of convenient dependency descriptors that you can use to bootstrap your spring apps.

AUTO CONFIGURATION:

Automatically configure Spring and 3rd party libraries/beans whenever possible

PROD READY FEATURES:

Inbuilt support of production-ready features such as metrics, health checks, and externalized configuration

SIMPLE CONFIGURATIONS :

Provides many annotations to do simple configurations and no requirement for XML configuration

ឧបសគ្គទី១នៃ Microservices ទំហំដែលត្រឹមត្រូវ និង កំណត់ព្រំដែននៃ Service

One of the most challenging aspects of building a successful microservices system is the identification of proper microservice boundaries and defining the size of each microservice.

Below are the most common followed approaches in the industry:


Domain-Driven Sizing - Since many of our modifications or enhancements driven by the business needs, we can size/define boundaries of our microservices that are closely aligned with Domain-Driven design & Business capabilities. But this process takes lot of time and need good domain knowledge.

Event Storming Sizing - Conducting an interactive fun session among various stake holder to identify the list of important events in the system like 'Completed Payment', 'Search for a Product' etc. Based on the events we can identify 'Commands', 'Reactions' and can try to group them to a domain-driven services.


Reference for Event Storming Session : <https://www.lucidchart.com/blog/ddd-event-storming>

Now let's take an example of a Bank application that needs to be migrated/build based on a microservices architecture and try to do sizing of the services.

1



Saving Account Trading Account



Card Loan

2





Saving Account Trading Account





Card Loan



3




Saving Account Trading Account





Credit Card Debit Card



Personal Loan Car Loan

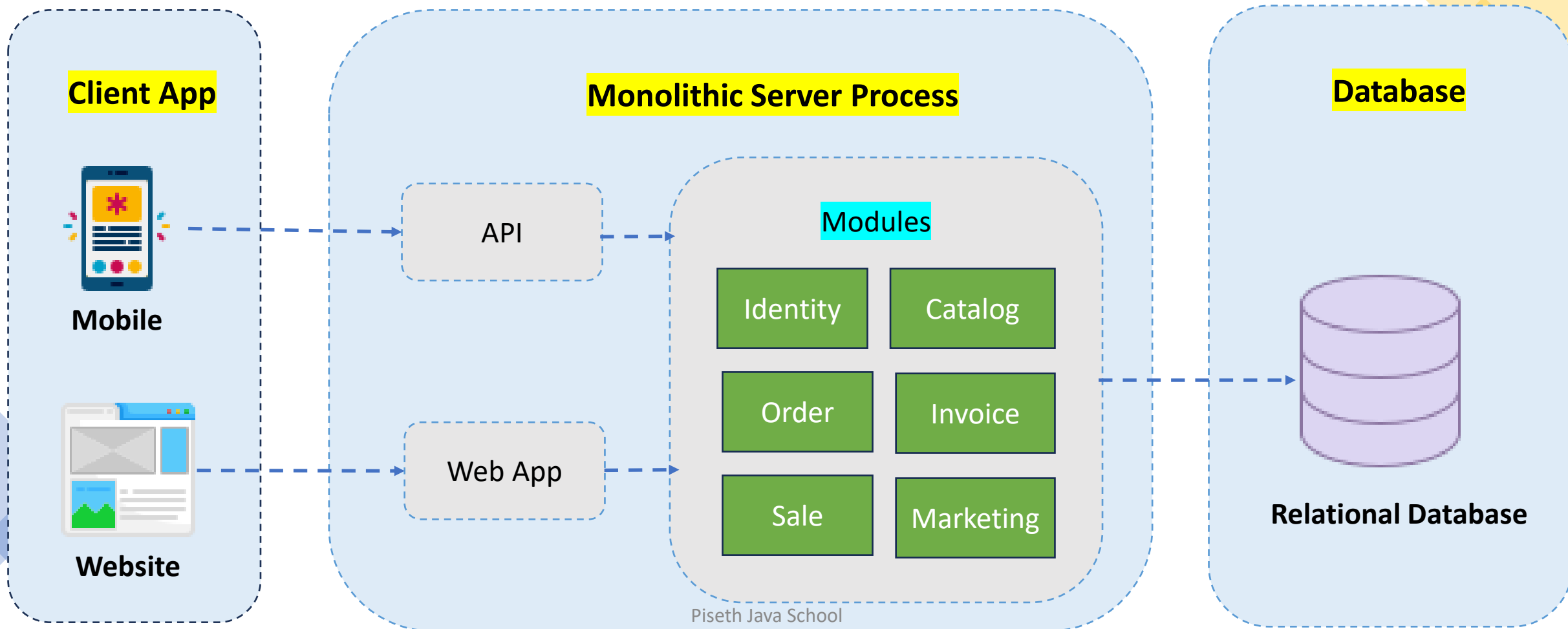


House Loan

- 
- 
1. NOT CORRECT SIZING AS WE CAN SEE INDEPENDENT MODULES LIKE CARDS & LOANS CLUBBED TOGETHER
 2. THIS MIGHT BE THE MOST REASONABLE CORRECT SIZING AS WE CAN SEE ALL INDEPENDENT MODULES HAVE SEPARATE SERVICE MAINTAINING LOOSELY COUPLED & HIGHLY COHESIVE
 3. NOT CORRECT SIZING AS WE CAN SEE TOO MANY SERVICES UNDER LOANS & CARDS

ការបំប្លែងពី Monolithic ទៅកាន់ Microservices

សូមពិនិត្យមើល scenario នៃ E-Commerce System ចាប់ផ្តើមឡើងដោយប្រើ monolithic architecture ហើយព្យាយាមស្វែងយល់ពីផលវិបាកទាំងឡាយនៃ architecture នេះ។



បញ្ហាដែល E-Commerce team កំពុងជួបប្រទះដោយសារ traditional monolithic design

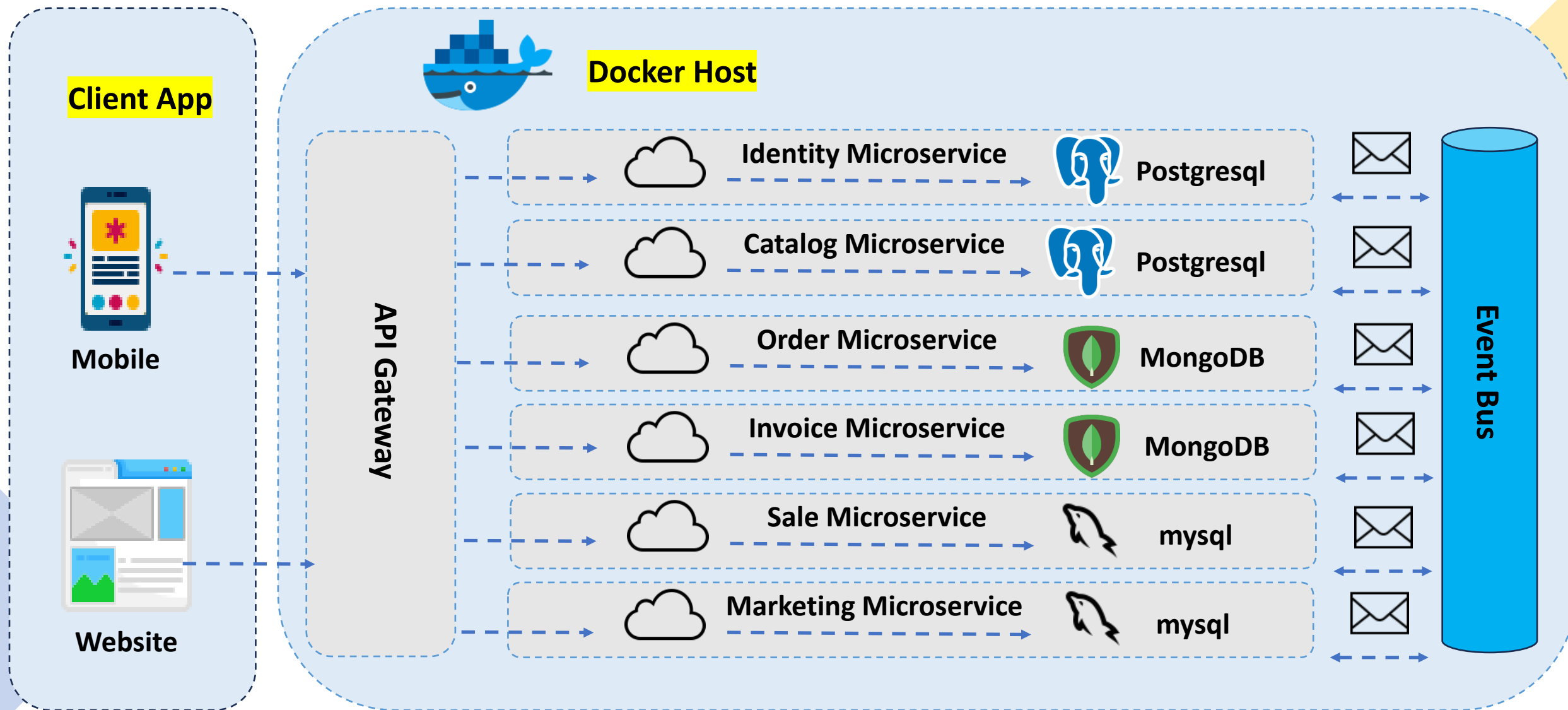
ថ្ងៃដំបូងៗ៖

វាងាយស្រួល build, test, deploy, troubleshoot និង scale កំឡុងពេល launch និងពេលដែលទំហំ Team នៅតូច។

បន្ទាប់ពីដាក់អោយប្រើប្រាស់មួយរយៈ app ឬ site ចាប់ផ្តើមលឿន ហើយមានអ្នកប្រើប្រាស់ច្រើន។ ហើយនេះជាបញ្ហា៖

- Application ប្រែទៅជាស្មុគស្មាញដែលគ្មានបុគ្គលណាម្នាក់អាចស្វែងយល់ដំណើរការទាំងមូល
- អ្នកមានការខ្លាចរអាក្នុងការកែប្រែ - ការកែប្រែដោយអចេតនា និងផលវិបាកក្រោយការ Update ចំណុចណាមួយ អាចប៉ះពាល់ផ្នែកដទៃ
- ចង់បន្ថែម feature ថ្មីឬការជួសជុល (Fix bug) ប្រើពេលយូរហើយចំណាយធនធានច្រើន
- Release ត្រឹមផ្នែកតូចមួយទាមទារអោយ Deploy Application ទាំងមូលឡើងវិញ
- ផ្នែកណាមួយដែលមានបញ្ហា វាអាចធ្វើអោយ system ទាំងមូលលែងដំណើរការ
- Technology ថ្មីៗចេញមក ក៏មិនអាចយកមកដាក់បញ្ចូលក្នុង System ដែលមានស្រាប់
- ពិបាកគ្រប់គ្រង Team តូចៗដែលកាន់ផ្នែកនីមួយៗ ហើយក៏ពិបាកយក Agile methodologies មកប្រើប្រាស់

E-Commerce ក្រោយពេលបំប្លែងទៅកាន់ Microservices



Microservices Architecture

- ងាយស្រួល Develop, Test និង Deploy Service នីមួយៗ
- ការអនុវត្តតាម Agile Methodology មានប្រសិទ្ធភាពខ្ពស់
- ងាយស្រួលក្នុងការ Scale Application (Horizontal Scale)
- Team ទាំងឡាយអាច Develop ដំណាលគ្នាបាន
- ការ Update ទៅលើ microservice ណាមួយមិនប៉ះពាល់ដល់ microservice ដទៃ
- Service នីមួយៗអាចជ្រើសរើស Language , Database ឬ Technology ផ្សេងគ្នាបាន។ល។