Section 1

Understanding Microservices

Instructor: Phạm Quang Anh Kiệt
@email: kietpham.dev@gmail.com
@facebook/rickykiet83

Introduction

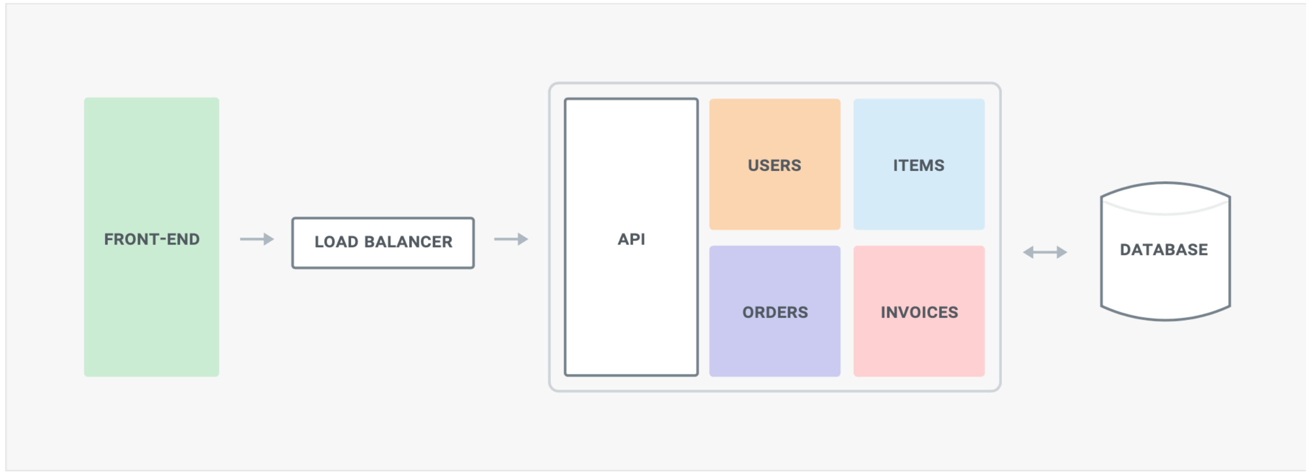
- Microservices is a hot trend in the technology section
- Netflix, Google, Twitter have been used microservices-based architecture
- It can be extremely daunting to start, however, for the larger enterprise, each modules can be developed with their own history and purpose

Advantages of Microservices

1. **Agility:** Componentization and distributed functionality empower developers to iterate and deploy continuously, autonomous of other business units and application teams.
2. **Freedom of Options**: Developers can independently pick their preferred framework (language, structure) to construct and convey functionality more rapidly.
3. **Resiliency**: Microservices are designed for failure with redundancy and isolation in mind, which in turn makes applications more robust.
4. **Efficiency**: There can be significant savings for the enterprise that de-couples functionality and adopts microservices.

Monolithic vs Microservices

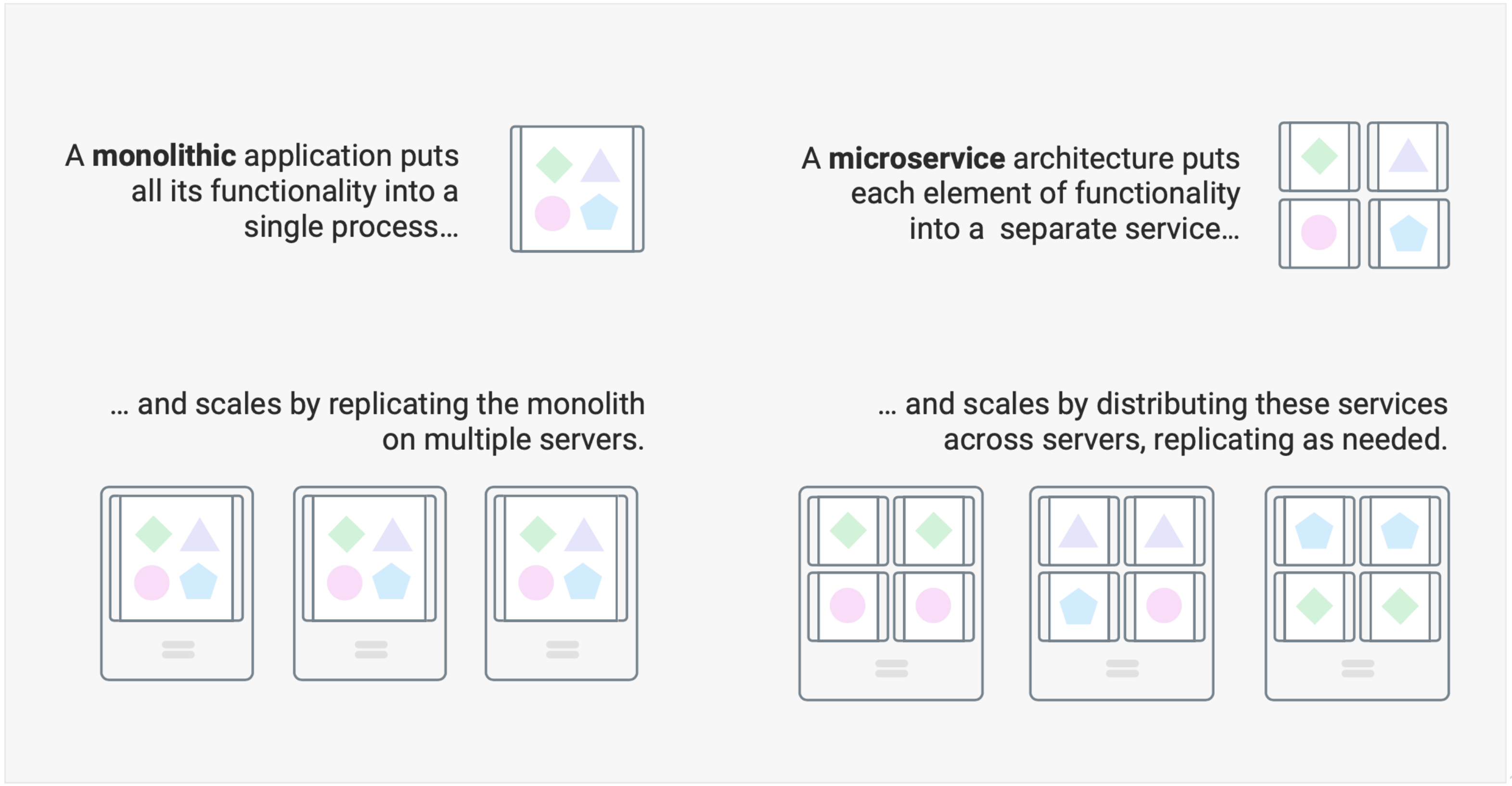
- Monolithic:
  - Easy to understand
  - It’s great when the codebase and the team working on it are both relatively small
  - A fast way to develop a product and get it into market quicky
  - No other dependencies.



Microservices

- Able to be **built independently**
- Able to be **deployed independently**
- Implementation detail will be taken care by the specific team working on that specific feature.
- Implementations of other components (services) work with interfaces, or APIs.
- One “big” specific thing tend to become much smaller => “microservices”

Microservices


A **monolithic** application puts all its functionality into a single process…

A **microservice** architecture puts each element of functionality into a separate service…

… and scales by replicating the monolith on multiple servers.

… and scales by distributing these services across servers, replicating as needed.

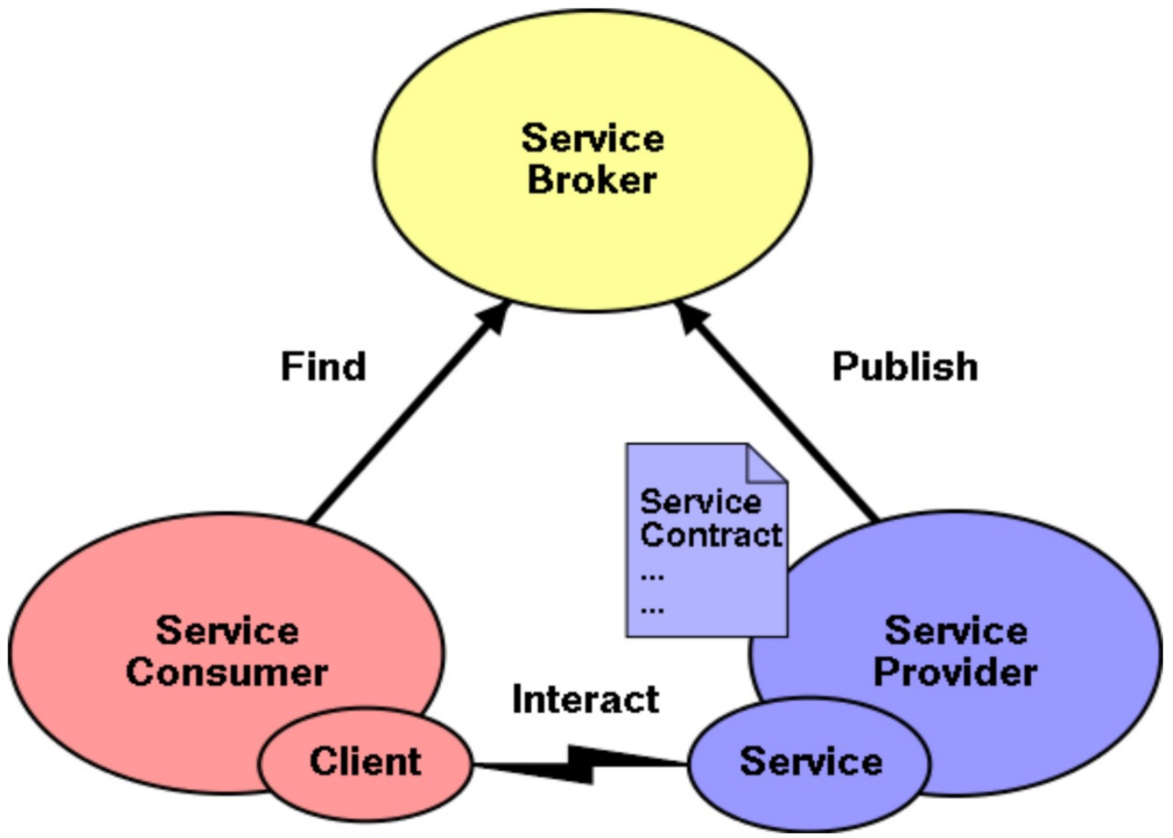
Microservices Pros and Cons

- Pros:
  - Better architecture for large applications
  - Better agility in the long term
  - Easy to learn
  - Isolation for scalability and damage control
- Cons:
  - More moving parts
  - Complex infrastructure requirements
  - Consistency and availability
  - Harder to test

Service-oriented architecture (SOA)

- *"Service-oriented architecture (SOA) is a type of software design that makes software components reusable using service interfaces that use a common communication language over a network."*
- In briefly, SOA integrates software components that have been separately deployed and maintained and allows them to communicate and work together to form software applications across different systems.

Service-oriented architecture (SOA)



Microservices architecture principles

1. **A microservices has a single concern.**
   - Should do one thing and one thing only = Single object responsibility
   - Easier to maintain and scale
2. A microservice is a discrete
   - Must clear boundaries separating it from its environment.
   - Must be well-encapsulated
   - Development: Isolated from all other microservices
   - Production: It becomes part of a larger application after deployment

Microservices architecture principles

3. **A microservices is transportable.**
   - Can be moved from one runtime environment to another
   - Easier to use in an automated or declarative deployment process.
4. A microservice carries its own data
   - Should have its own data storage that is isolated from all other microservices.
   - Shared with other microservices by a public interface
   - The common problem is data redundancy.

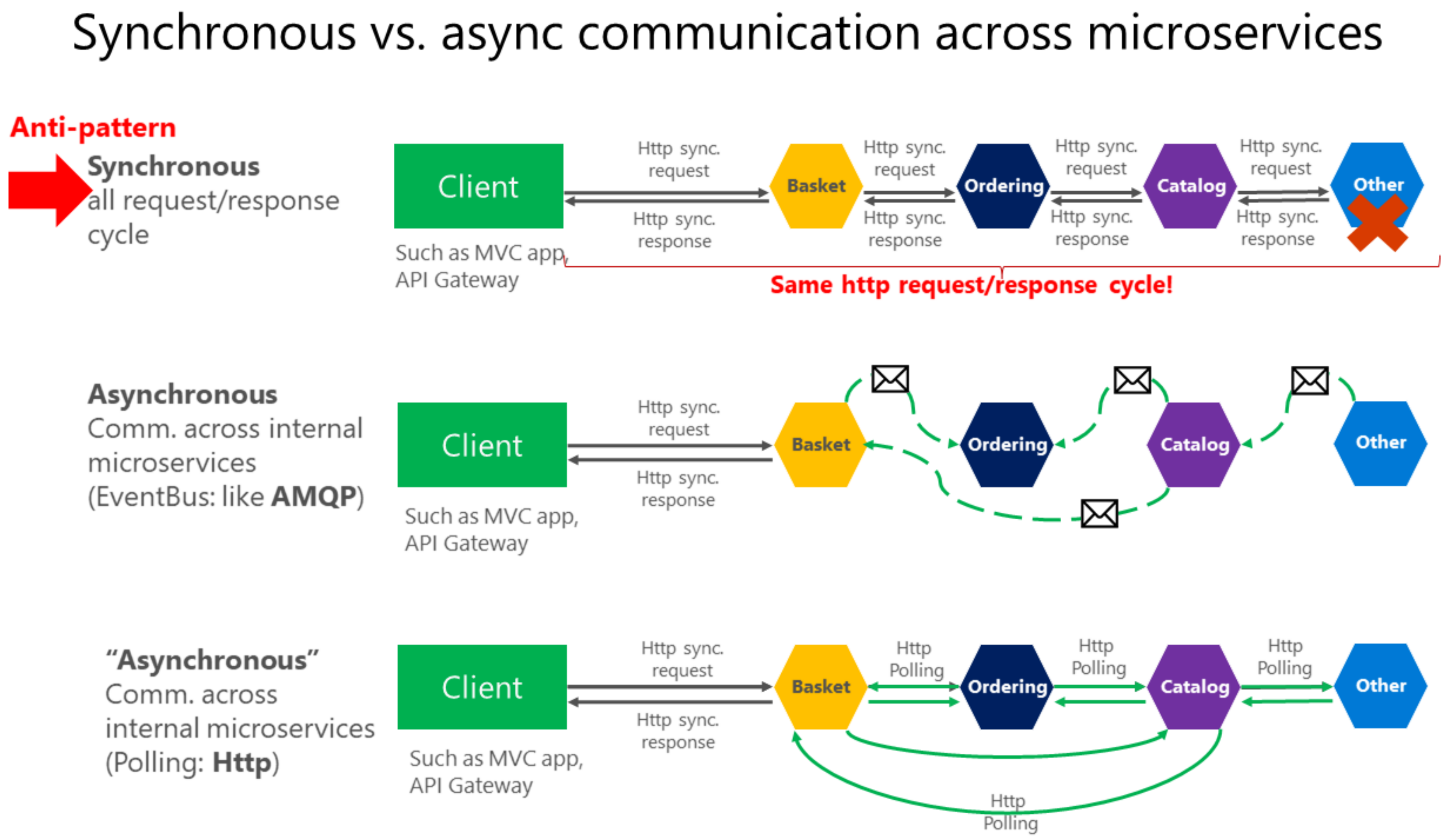
Microservices architecture principles

5. **A microservice is ephemeral**
   - It can be created, destroyed, and replenished on demand
   - The standard operating expectation is that microservices come and go all the time, sometimes due to system failure and sometimes due to scaling demands.

Microservice communication

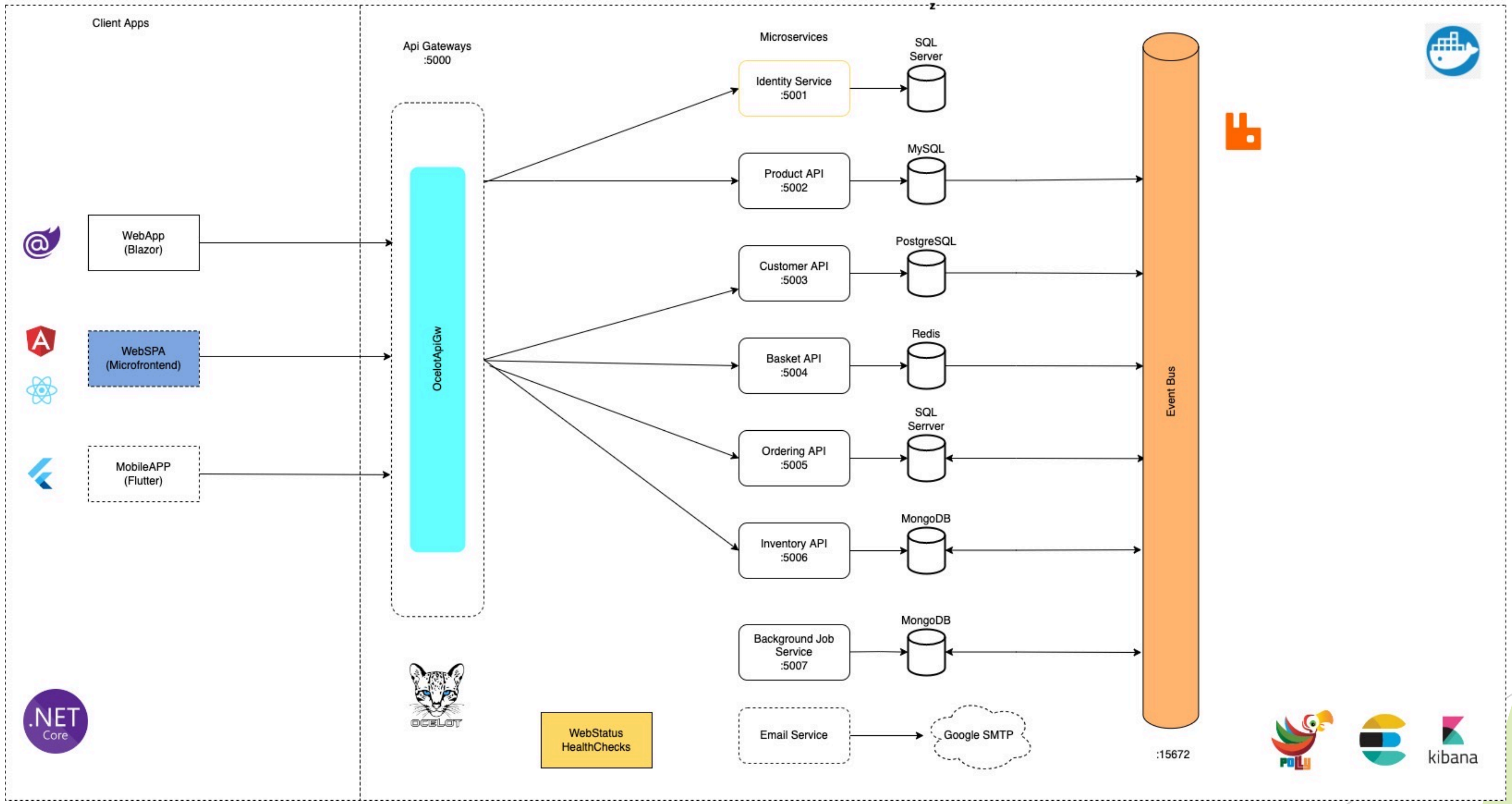
1. Synchronous protocol
   - HTTP/HTTPS
   - The client sends a request and waits for a response from the service
   - Thread is blocked
   - The client code can only continue its task when it receives the HTTP server response.
2. Asynchronous protocol
   - AMQP (a protocol supported by many OS and cloud environments)
   - Asynchronous messages
   - The client send message and doesn’t wait for a response.
   - RabbitMQ or Kafka is a message queque

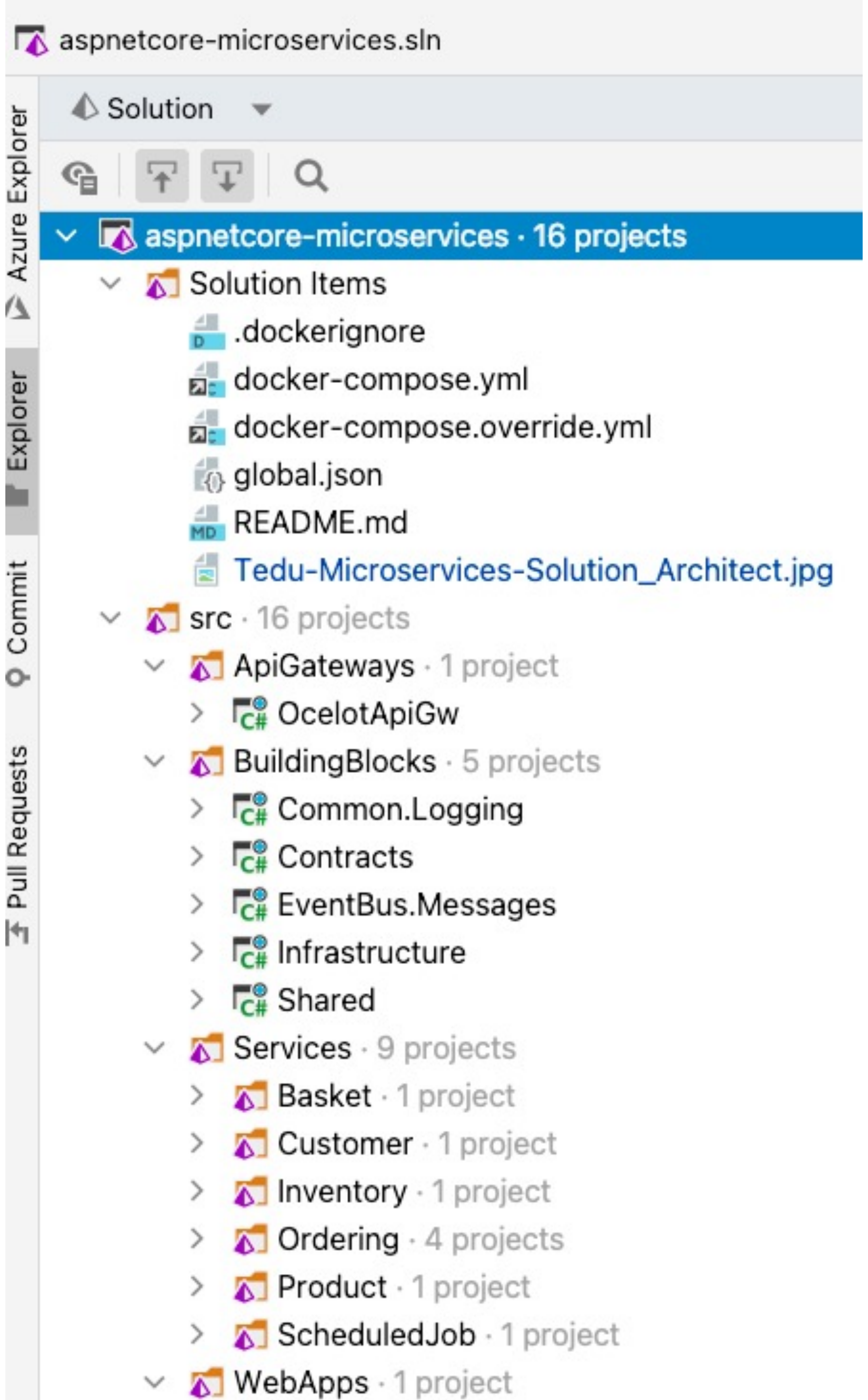
Microservice communication



Synchronous vs. async communication across microservices
Anti-pattern
Synchronous
all request/response cycle
Client
Such as MVC app, API Gateway
Http sync. request
Http sync. response
Basket
Ordering
Catalog
Other
Same http request/response cycle!
Asynchronous
Comm. across internal microservices
(EventBus: like AMQP)
"Asynchronous"
Comm. across internal microservices
(Polling: Http)
Http Polling

Tedu aspnetcore Microservices project



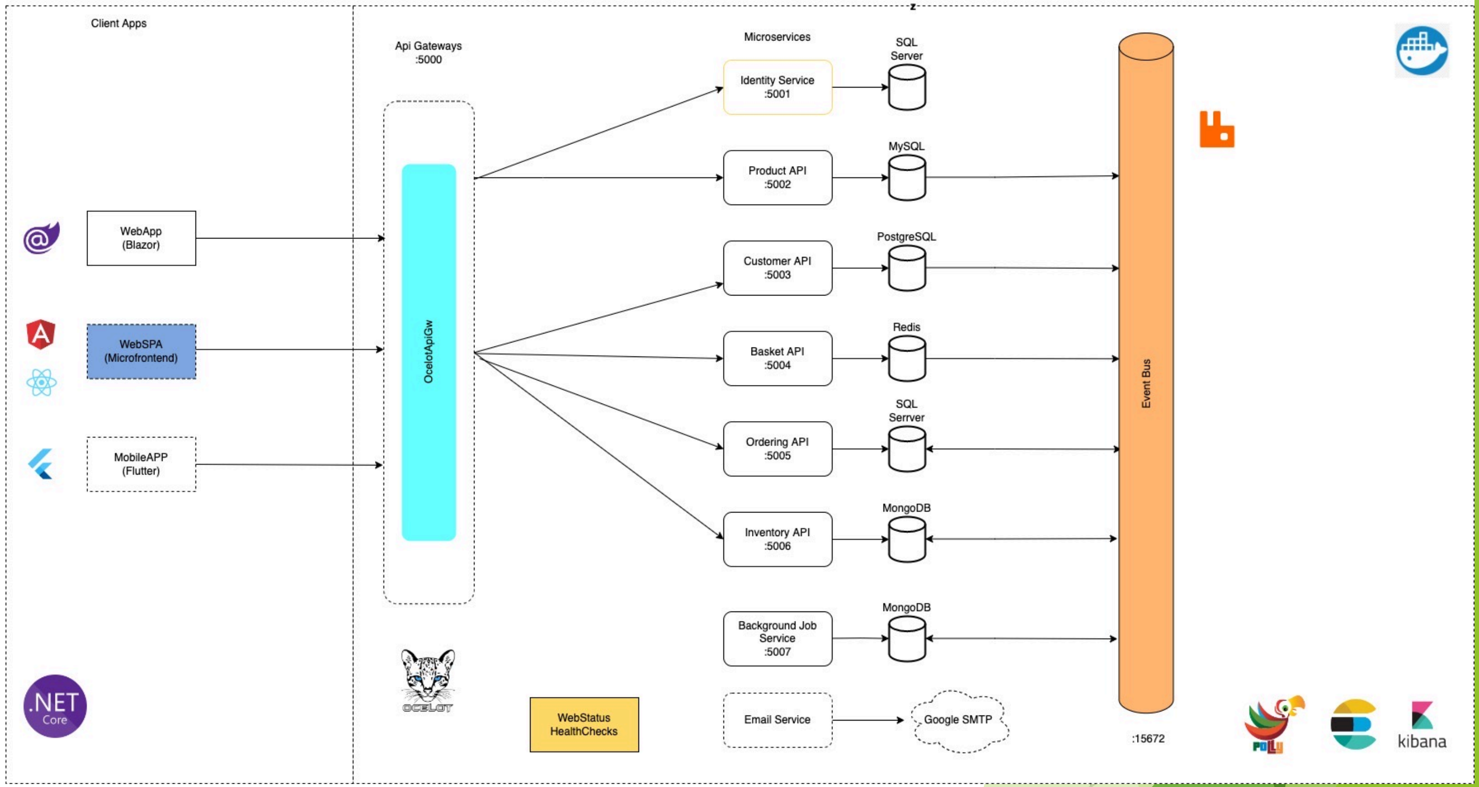


aspnetcore-microservices.sln

Azure Explorer | Explorer | Commit | Pull Requests

Solution

- aspnetcore-microservices · 16 projects
  - Solution Items
    - .dockerignore
    - docker-compose.yml
    - docker-compose.override.yml
    - global.json
    - README.md
    - Tedu-Microservices-Solution_Architect.jpg
  - src · 16 projects
    - ApiGateways · 1 project
      - OcelotApiGw
    - BuildingBlocks · 5 projects
      - Common.Logging
      - Contracts
      - EventBus.Messages
      - Infrastructure
      - Shared
    - Services · 9 projects
      - Basket · 1 project
      - Customer · 1 project
      - Inventory · 1 project
      - Ordering · 4 projects
      - Product · 1 project
      - ScheduledJob · 1 project
    - WebApps · 1 project



Solution exploration

- **Building Blocks**: Including class libraries which defines interfaces, contracts, shared and common methods.
  - Common.Logging: Logging system with Serilog and elasticsearch.
  - Contracts: The blue print of the system, where we can define the common interfaces as: Repository, UnitOfWork… to define our contracts for the whole system.
  - EventBus.Message: Event Bus Message system, AMQP, standardize communication across microservices.
  - Infrastructure: Class library implements from Contracts interface.
  - Shared: Sharing resources, common variables, configurations across microservices.

Solution exploration

- Services: Including the microservices of the system.
  - Basket: Basket API with Redis
  - Customer: Customer Minimal API with PostgreSQL
  - Ordering: Ordering API with Clean Architecture and SQL Server
  - Product: Product API with MySQL
  - Inventory: Inventory API with MongoDB
  - ScheduledJob: Hangfire API with MongoDB, background tasks

Solution exploration

- WebApps:
  - WebHealthStatus MVC, presentation health check system.
  - Microfrontend Client App (not included in this course)

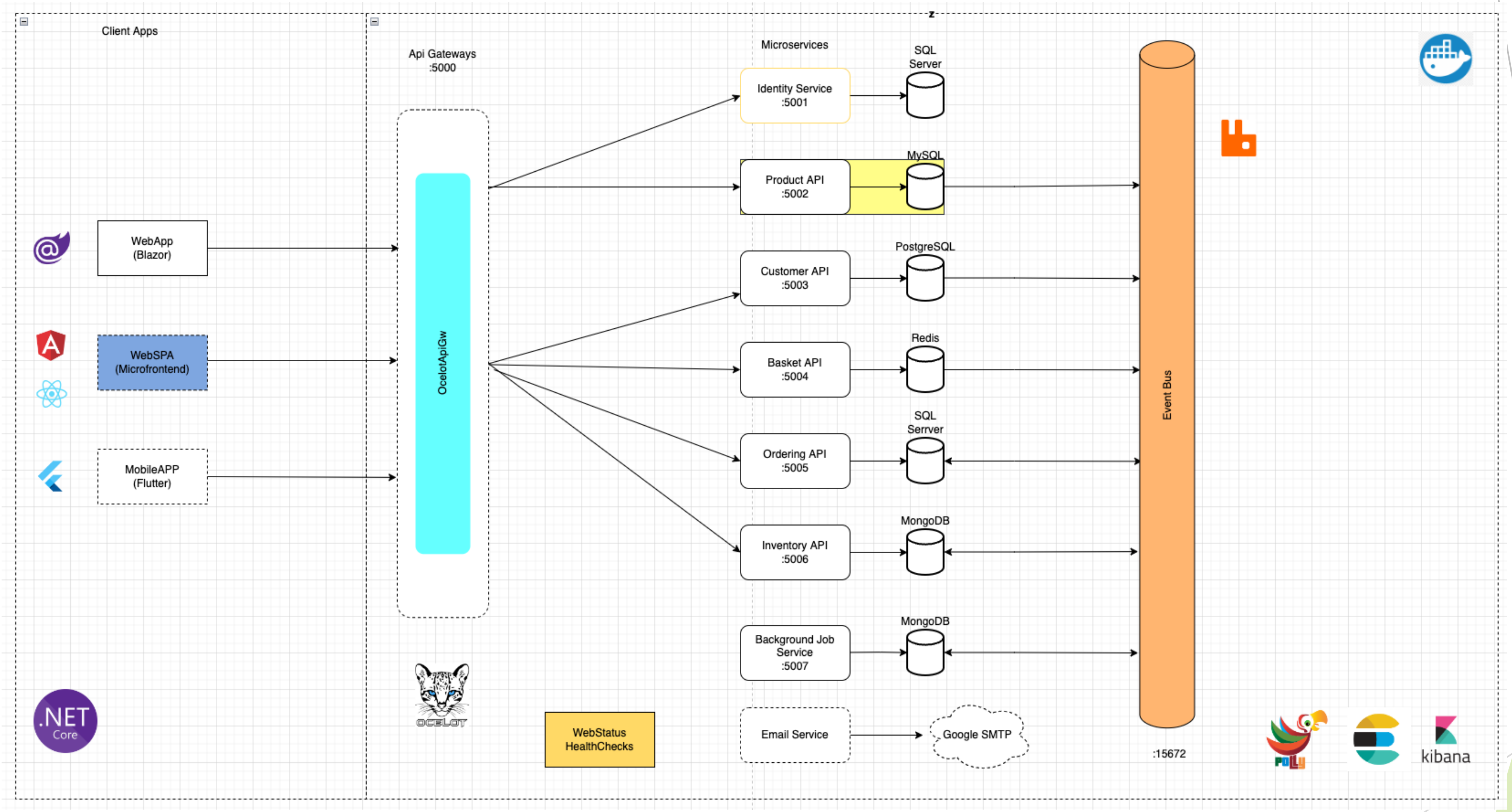
Section 2

Product API with MySQL

Instructor: Phạm Quang Anh Kiệt

@email: kietpham.dev@gmail.com

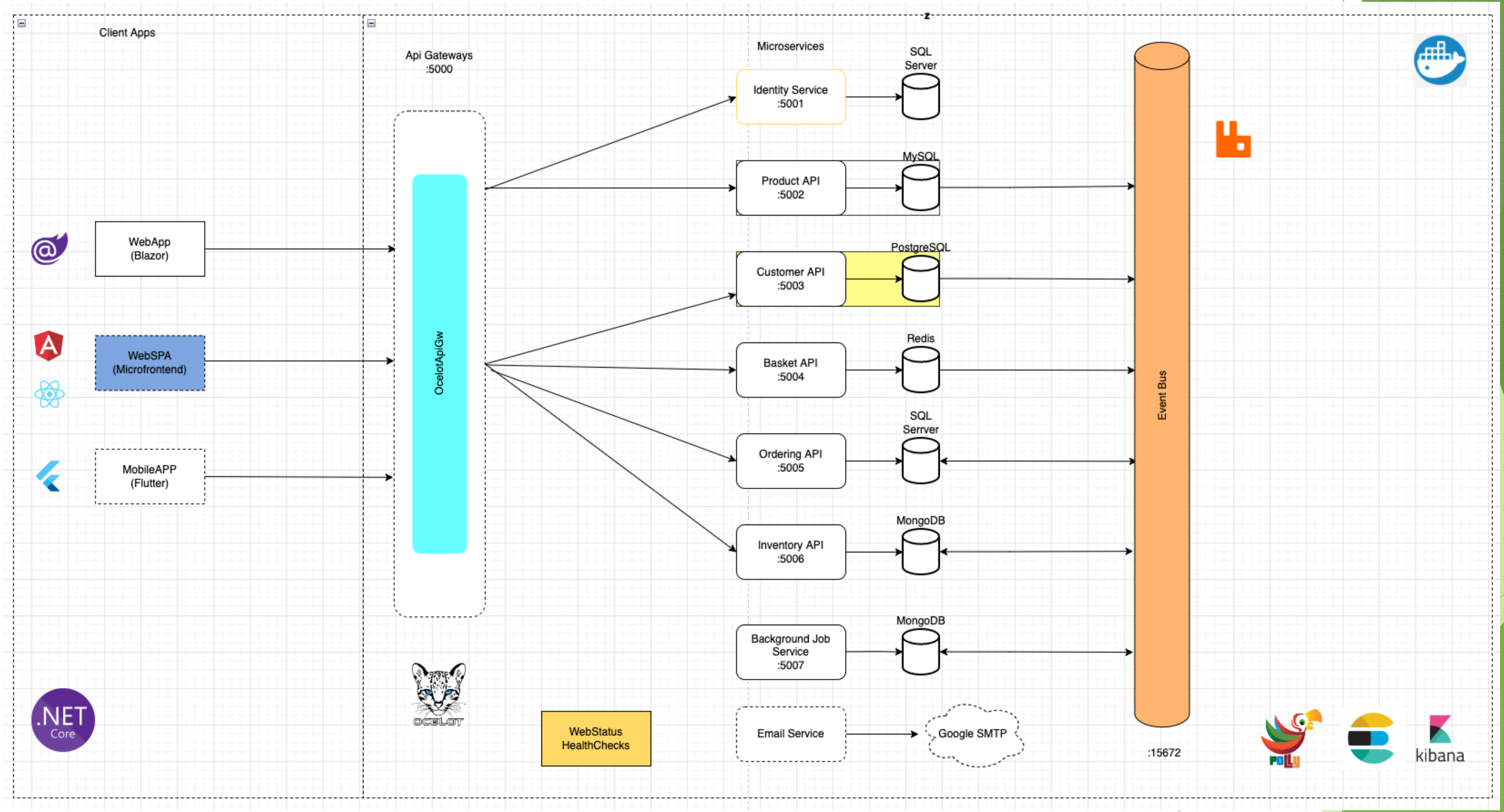
@facebook/rickykiet83


Client Apps
Api Gateways
:5000
Microservices
WebApp
(Blazor)
WebSPA
(Microfrontend)
MobileAPP
(Flutter)
OcelotApiGw
Identity Service
:5001
SQL
Server
Product API
:5002
MySQL
Customer API
:5003
PostgreSQL
Basket API
:5004
Redis
Ordering API
:5005
SQL
Serrver
Inventory API
:5006
MongoDB
Background Job
Service
:5007
MongoDB
Email Service
Google SMTP
WebStatus
HealthChecks
Event Bus
:15672
.NET
Core
OCELOT
POLLY
kibana

Section 3

Customer API with Minimal API & PostgreSQL

Instructor: Phạm Quang Anh Kiệt
@email: kietpham.dev@gmail.com
@facebook/rickykiet83

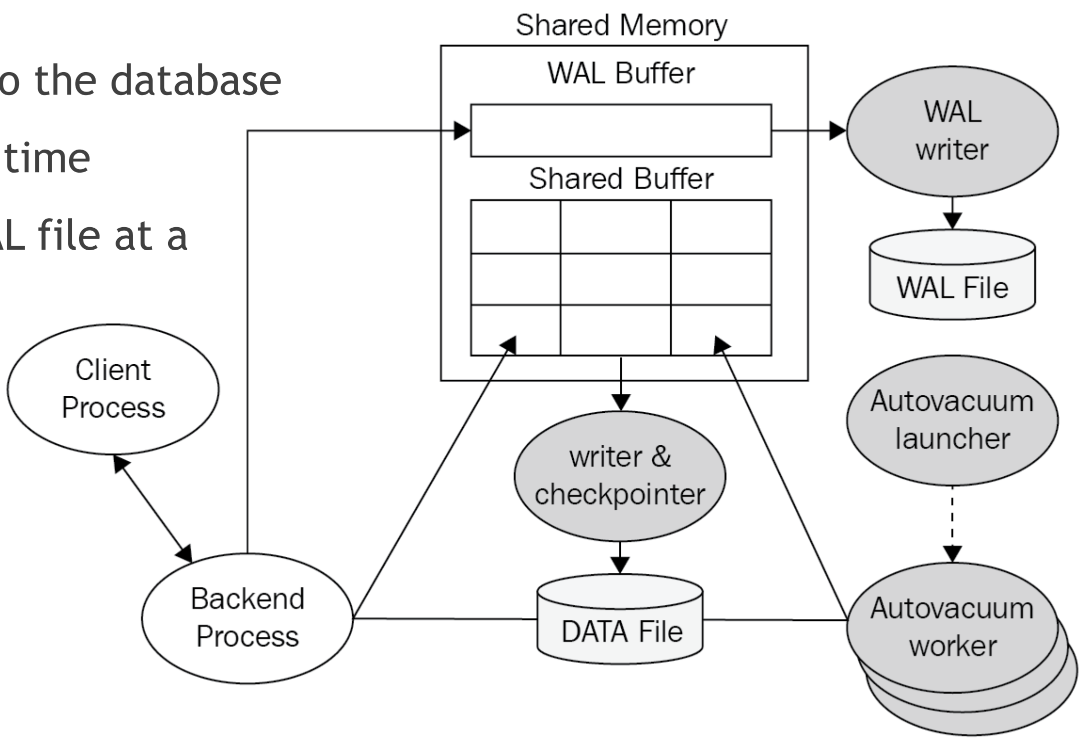

Client Apps
WebApp (Blazor)
WebSPA (Microfrontend)
MobileAPP (Flutter)
.NET Core
Api Gateways :5000
OcelotApiGw
OCELOT
WebStatus HealthChecks
Microservices
Identity Service :5001
Product API :5002
Customer API :5003
Basket API :5004
Ordering API :5005
Inventory API :5006
Background Job Service :5007
Email Service
SQL Server
MySQL
PostgreSQL
Redis
SQL Serrver
MongoDB
MongoDB
Google SMTP
Event Bus
:15672
kibana

An overview of PostgreSQL

- Released in 1994
- PostgreSQL is an **object-relational database management system** (**ORDBMS**)
- A popular **Database as a Service** (**DBaaS**)
- PostgreSQL can be delivered as DBaaS on many clouds, such as **Amazon Web Services** (**AWS**), Google Cloud SQL, Microsoft Azure, Heroku, and EnterpriseDB Cloud.
- Open source **license is free**, so developers can easily operate as many databases as they wish without any cost.

The PostgreSQL architecture

- Shared memory:
  - Minimize DISK I/O
  - Access very large buffers (tens or hundreds of gigabytes) worth) quickly.
  - The reduction of **write-ahead log** (**WAL**) (Nhật ký ghi trước)
  - The WAL buffer is a buffer that temporarily stores changes to the database
  - Minimize contention when many users access it at the same time
  - The contents stored in the WAL buffer are written to the WAL file at a predetermined point in time.

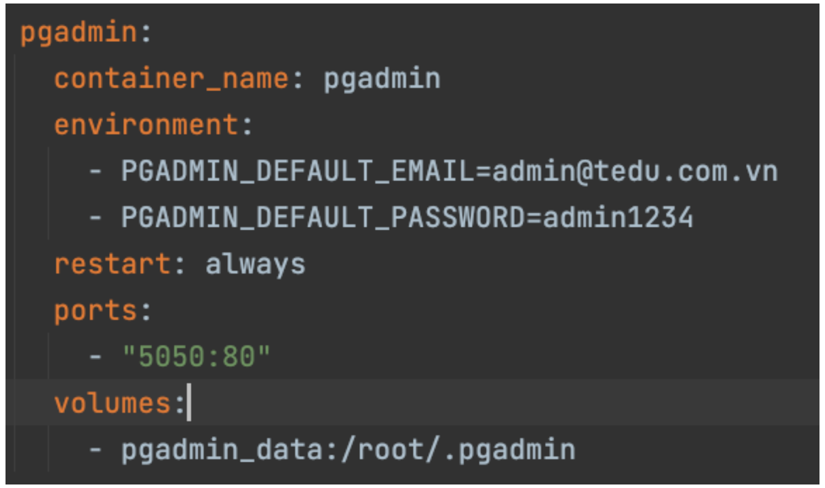


Standout Features:

- Complex query
- Trigger
- View
- Integrity transactions
- Multi-version concurrency control (Kiểm tra truy cập đồng thời đa phiên bản)
- Parallel query
- Types: JSON/JSONB, XML, Key-Value
- Point-in-time-recovery – PITR)
- Authentication: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, Certificate
- Columns/Rows security
- Index: B-tree, Multicolumn, Expression, Partial
- Advanced Index: GiST, SP-Gist, KNN Gist, GIN, BRIN, Bloom filters

PgAdmin4

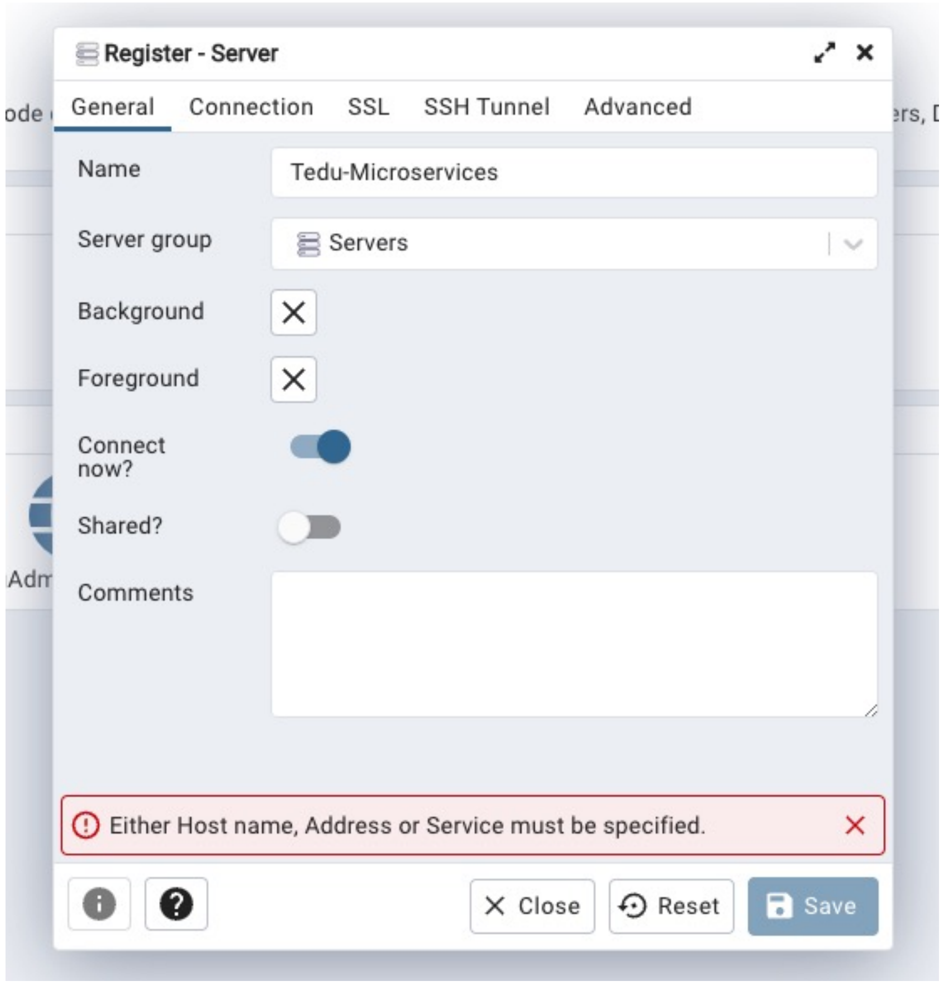
- https://www.pgadmin.org/
- Download desktop version at: https://www.pgadmin.org/download/
- Or using docker at: http://localhost:5050 (docker-compose-override.yml file)



```
pgadmin:
  container_name: pgadmin
  environment:
    - PGADMIN_DEFAULT_EMAIL=admin@tedu.com.vn
    - PGADMIN_DEFAULT_PASSWORD=admin1234
  restart: always
  ports:
    - "5050:80"
  volumes:
    - pgadmin_data:/root/.pgadmin
```

PgAdmin4

- Add new Server



Register - Server

Name: Tedu-Microservices

Server group: Servers

Background

Foreground

Connect now?

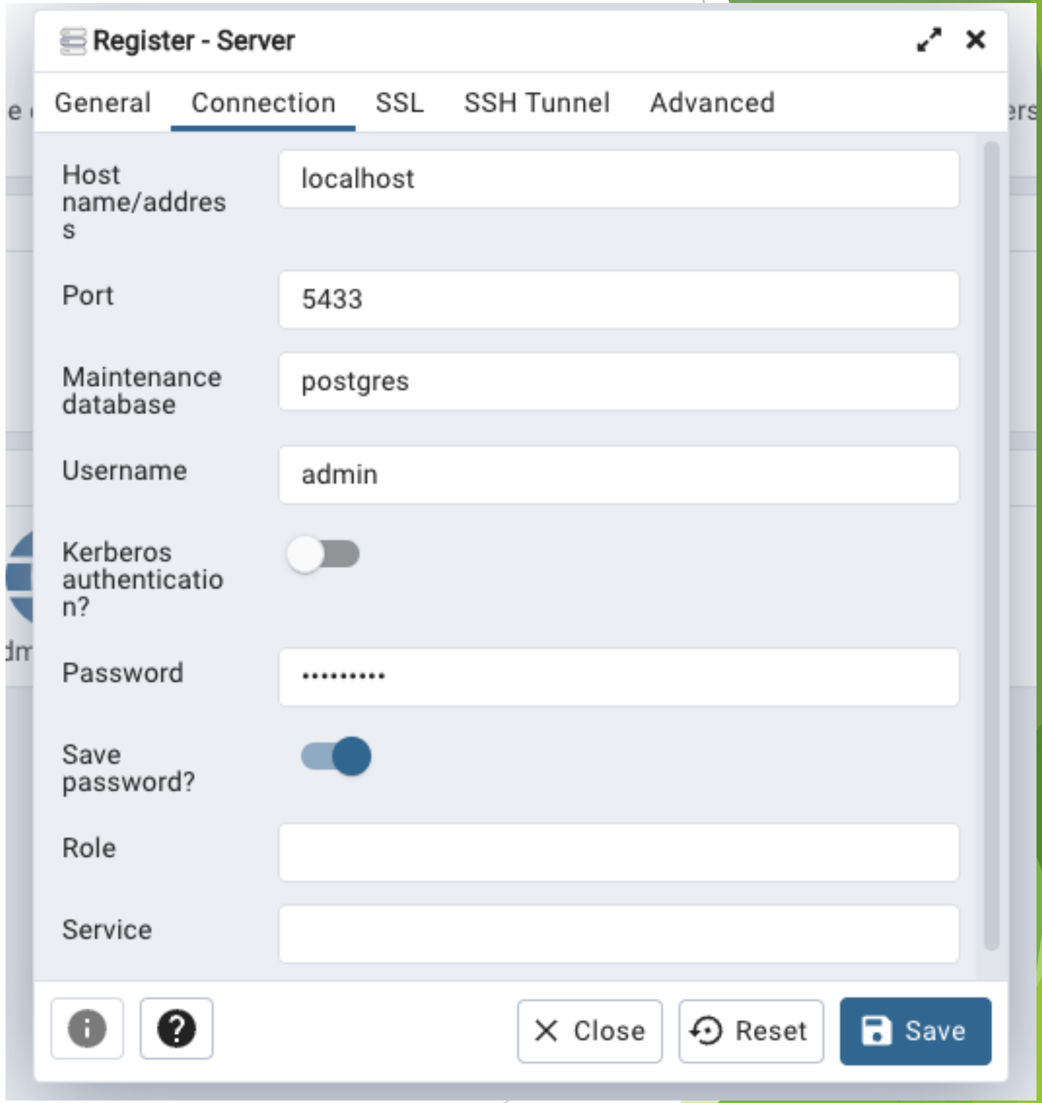
Shared?

Comments

Either Host name, Address or Service must be specified.

PgAdmin4

- Connection (docker-compose.override.yml)
- Username: admin
- Password: admin1234


Register - Server

General | Connection | SSL | SSH Tunnel | Advanced

Host name/address	localhost
Port	5433
Maintenance database	postgres
Username	admin
Kerberos authentication?	off
Password	••••••••••
Save password?	on
Role	
Service	