Understanding Microservices Architecture

Maaike van Putten
Software developer & trainer
www.biltup.com

Y

Session 1

Introduction to microservices

d

Session 2

Comparing microservices and monolithic architecture

d

Session 3

Building and containerizing microservices

E

Session 4

Managing data in microservices

Introduction to Microservices

Learning objectives

- Define **what microservices are** and identify the **core characteristics** that differentiate them from traditional monolithic applications.
- **Understand the benefits of microservices** such as improved modularity, scalability, and isolation, making them suitable for complex, evolving applications.

Schedule

Intro + theory

Introduction round + what are microservices

b

Exercise

Analyze and deconstruct a microservice application

D

Debrief + theory

Debrief exercise + Microservice terminology

b

Exercise

Design a microservices architecture

q

Debrief + wrap up

Debrief exercise + mini quiz

Introduction round

 Current role, background and ambitions

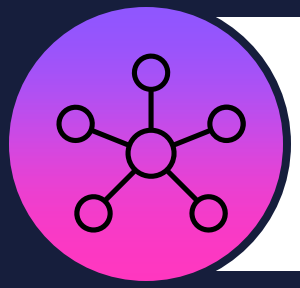
 Hobbies / interests / life outside of work

 What do you hope to learn?

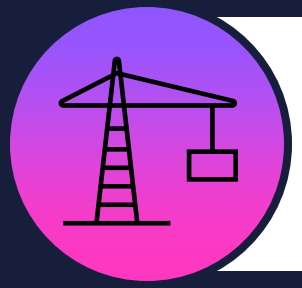
Microservices

- P — Software development practices
- m — Increase development speed
- d — Scaling up
- k — Not bound to a certain technology
- . — Principles and patterns

Service



Service Oriented Architecture (SOA)



Deployable independently

d

Independent component

S

Message based communications

b

Organized around business capability

I

Owned by a small(er) team

When is a service micro?

I

No agreed size

R

Common rule: one responsibility / functionality

C

Bounded context per microservice: all terms and entities have a clear and unambiguous meaning inside the microservice

Componentization via Services

d

Component

Self-contained unit of software, independently replaceable and upgradable

Services vs. libraries

Services are independently deployable, unlike libraries

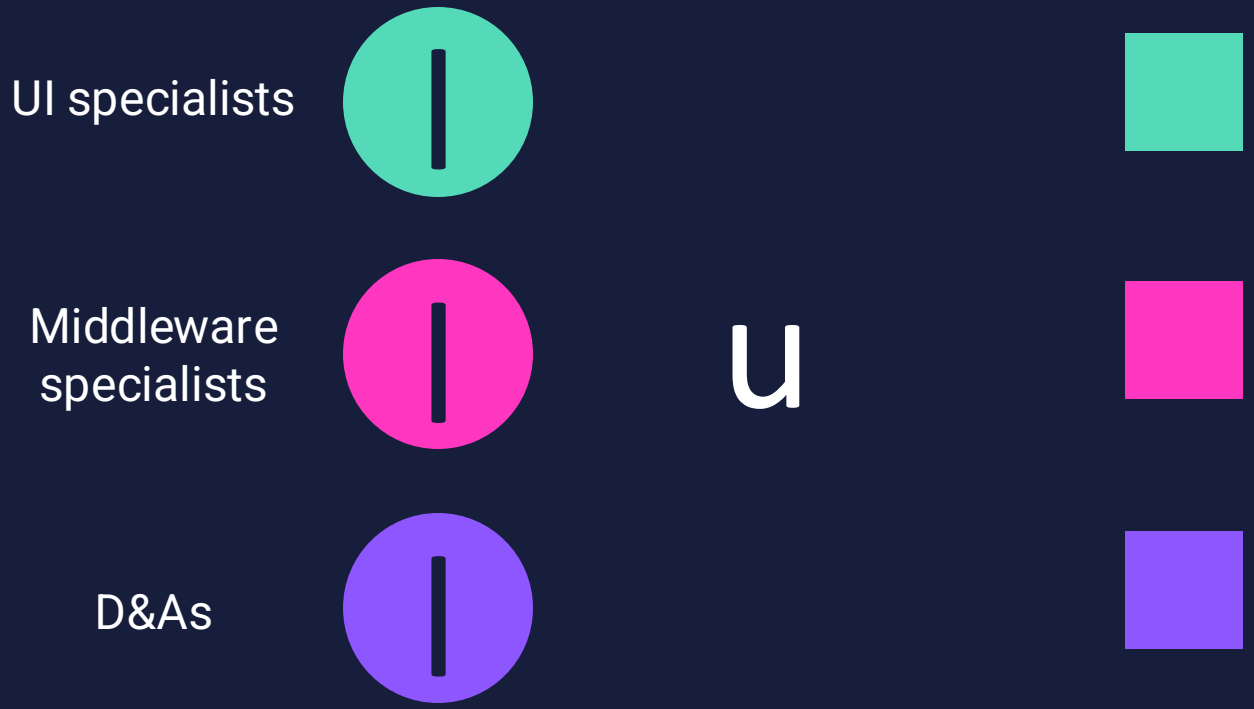
o

Communication

Services interact via well-defined public APIs

Organized around business capabilities

Siloed teams... lead to siloed application structure



Organized around business capabilities

Siloed teams... lead to siloed application structure

UI specialists

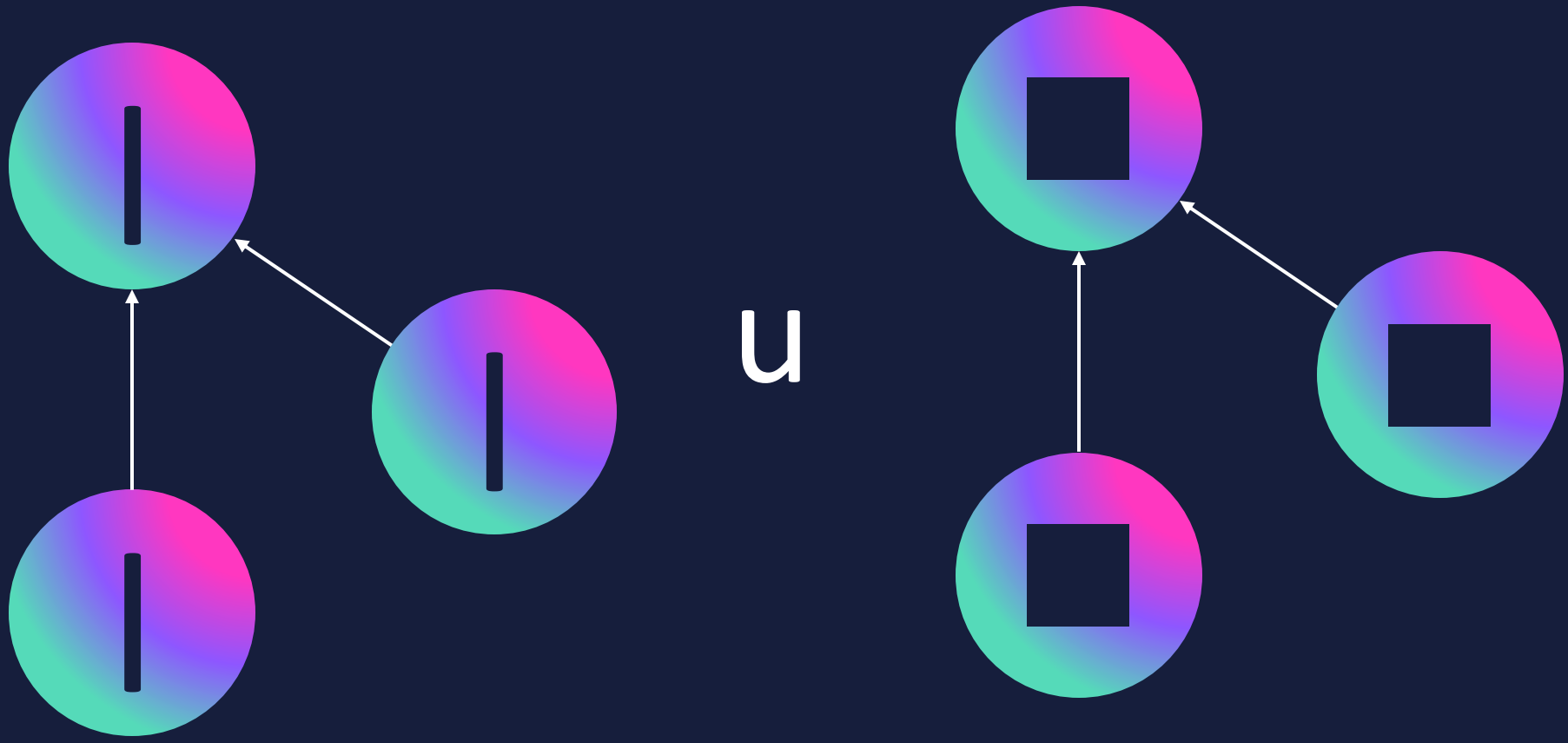
Middleware specialists

D&As

u

Organized around business capabilities

Cross functional teams... organized around capabilities



Smart design

- Decoupled and cohesive applications
- a — Receiving requests, process them and send back responses
- T — Simple communication, e.g. REST instead of complex protocols
- w — Basic infrastructure

Decentralized governance

- Different problems need different solutions
- Use the right tools
- Pick the best technology or language for each service

Software development lifecycle

- e — Analysis & requirements
- N — Design
- h — Implement
- I — Test
- A — Deploy
- Y — Maintenance and support

r

For a big system the TTM is longer than for a small system

k

Microservices allow more agile approach and give teams more freedom

Exercise

Analyze and deconstruct a microservice application

https://github.com/FudanSELab/train-ticket/

Microservice terminology

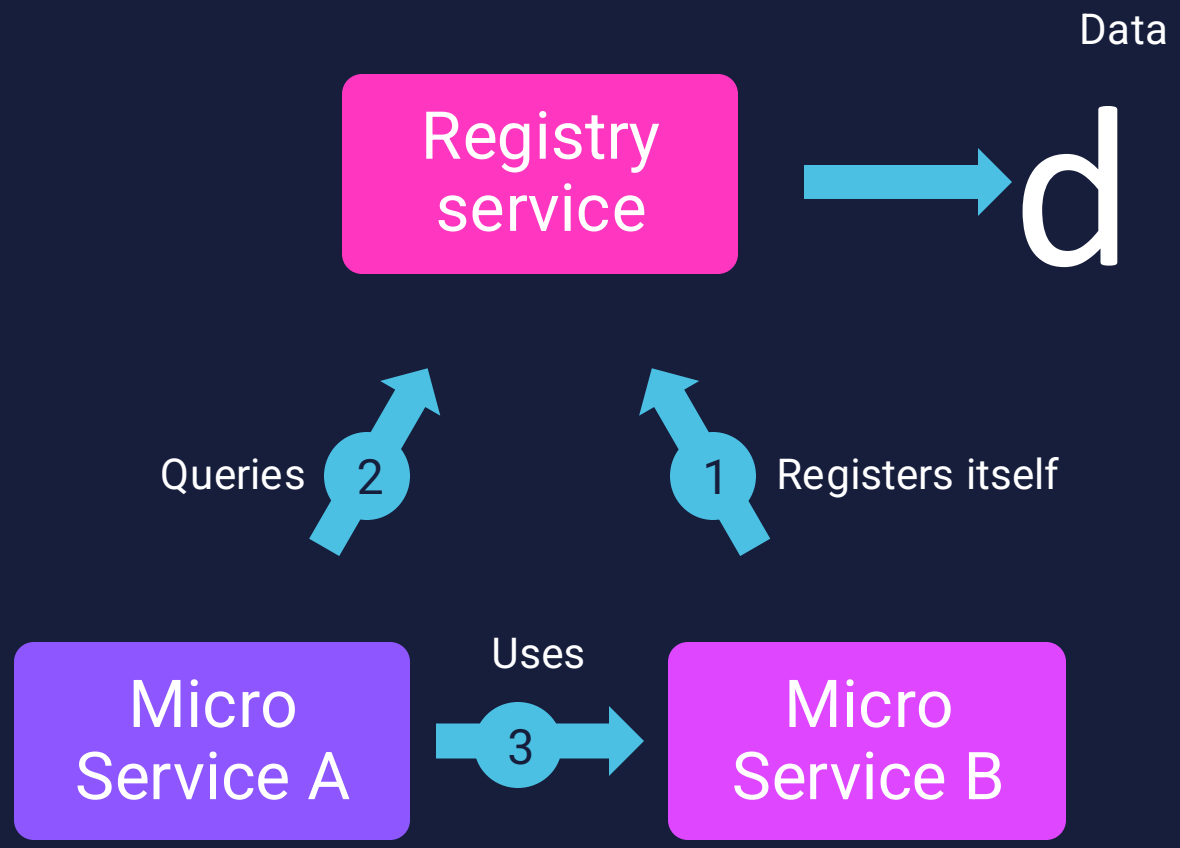
Load balancer

Distributes incoming traffic evenly across multiple microservice instances

Microservice terminology

Service Discovery server

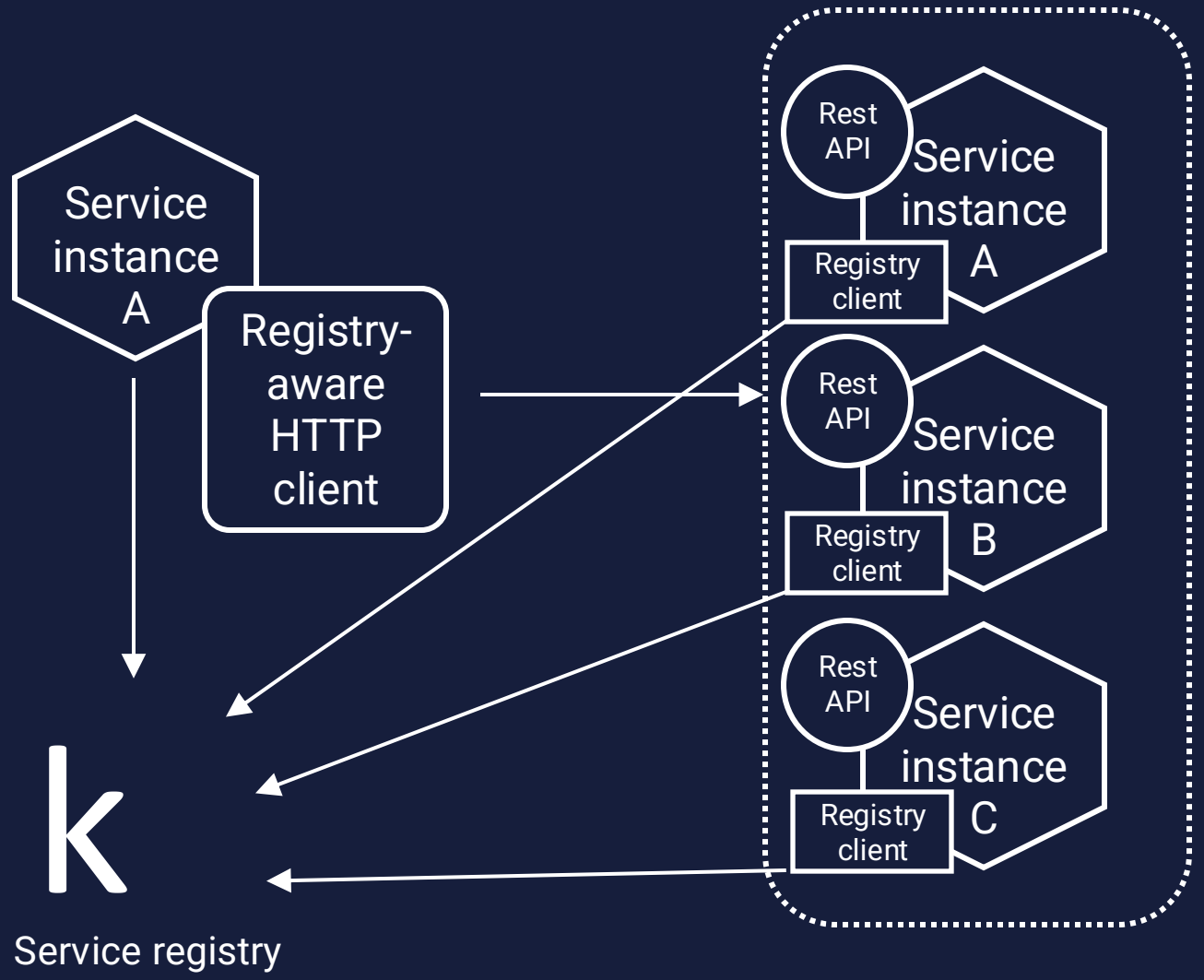
- Tracking deployed microservices and their locations can be challenging
- Service discovery solution
- Allows microservices to automatically register at startup
- Through an API



Client vs server-side service discovery

Client-side discovery

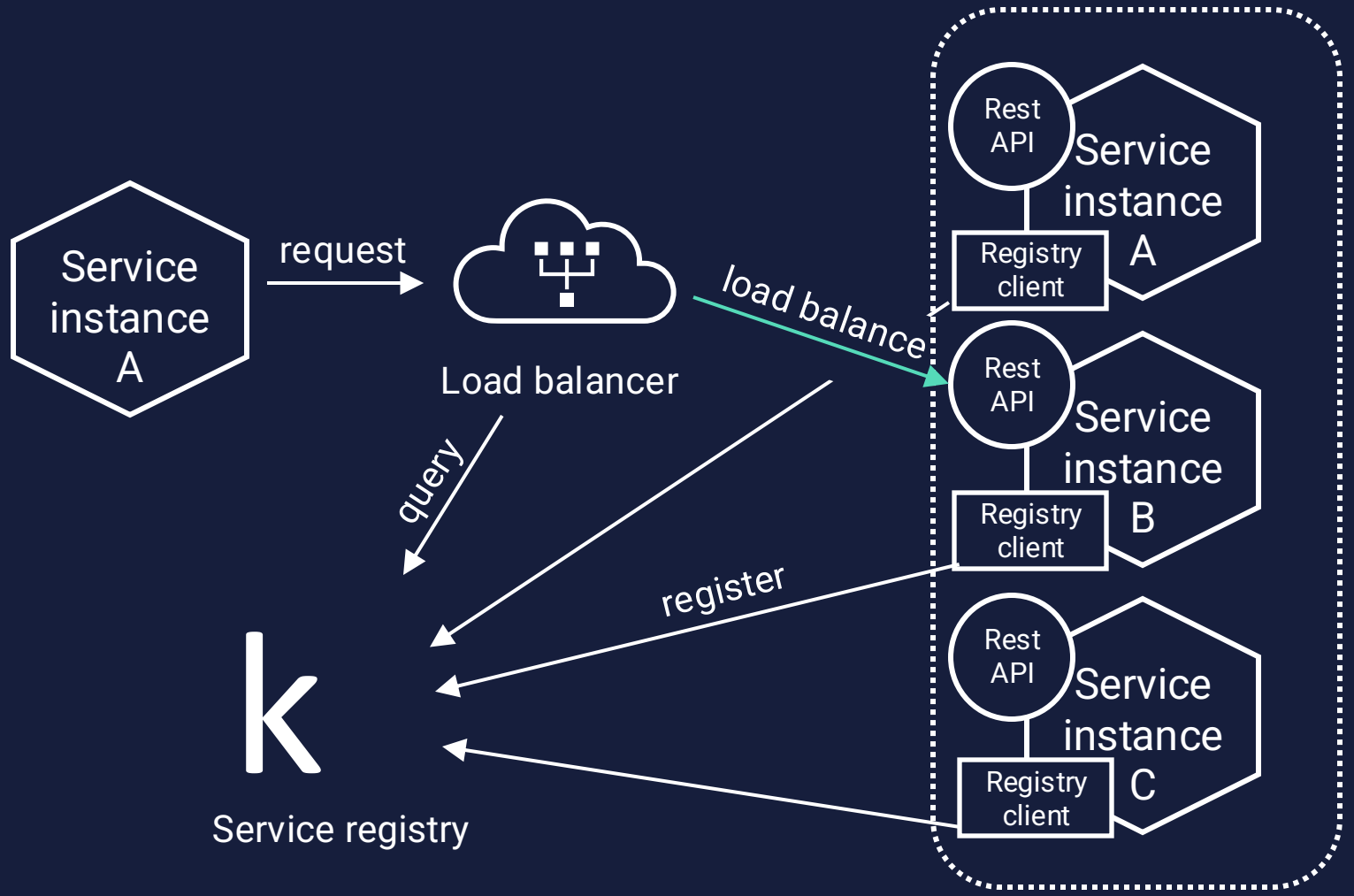
- The client communicates **directly** with the **service registry** and handles load balancing
- The client needs to be aware of the service registry



Client vs server-side service discovery

Server-side discovery

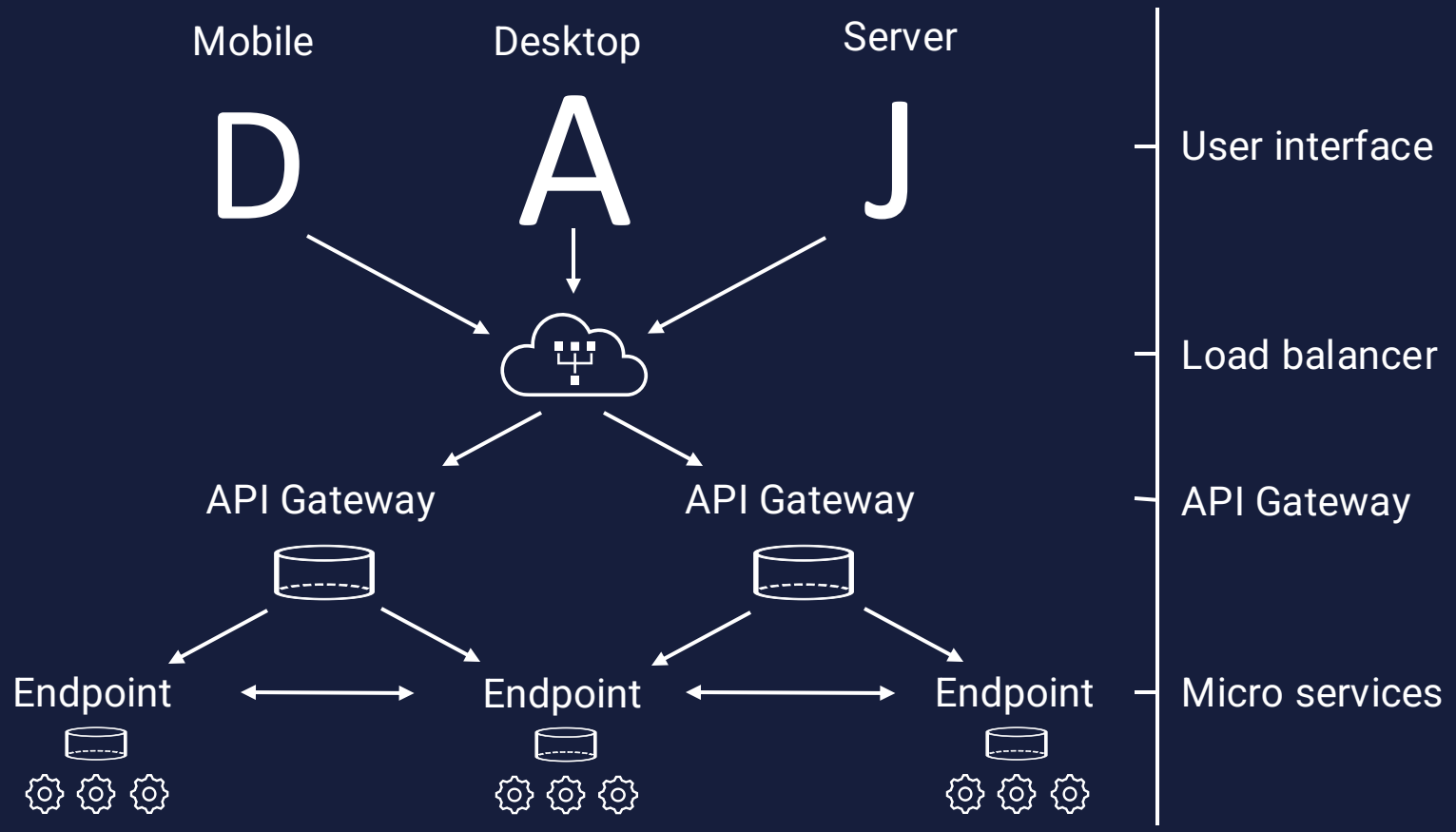
- The client talks to a **load balancer**, which in turn interacts with the **service registry**
- The client does not need to know about the service registry



Microservice terminology

API Gateway

- Shields clients from the complexity of how the application is divided into microservices
- Hides the challenge of locating service instances from the clients
- Delivers the most suitable API for each client



Microservice terminology

Central configuration server

Centralized configuration management replaces local configurations

Monitoring

Essential as the number of microservices grow

Containerization

Containers simplify managing and updating services

Microservice terminology

e

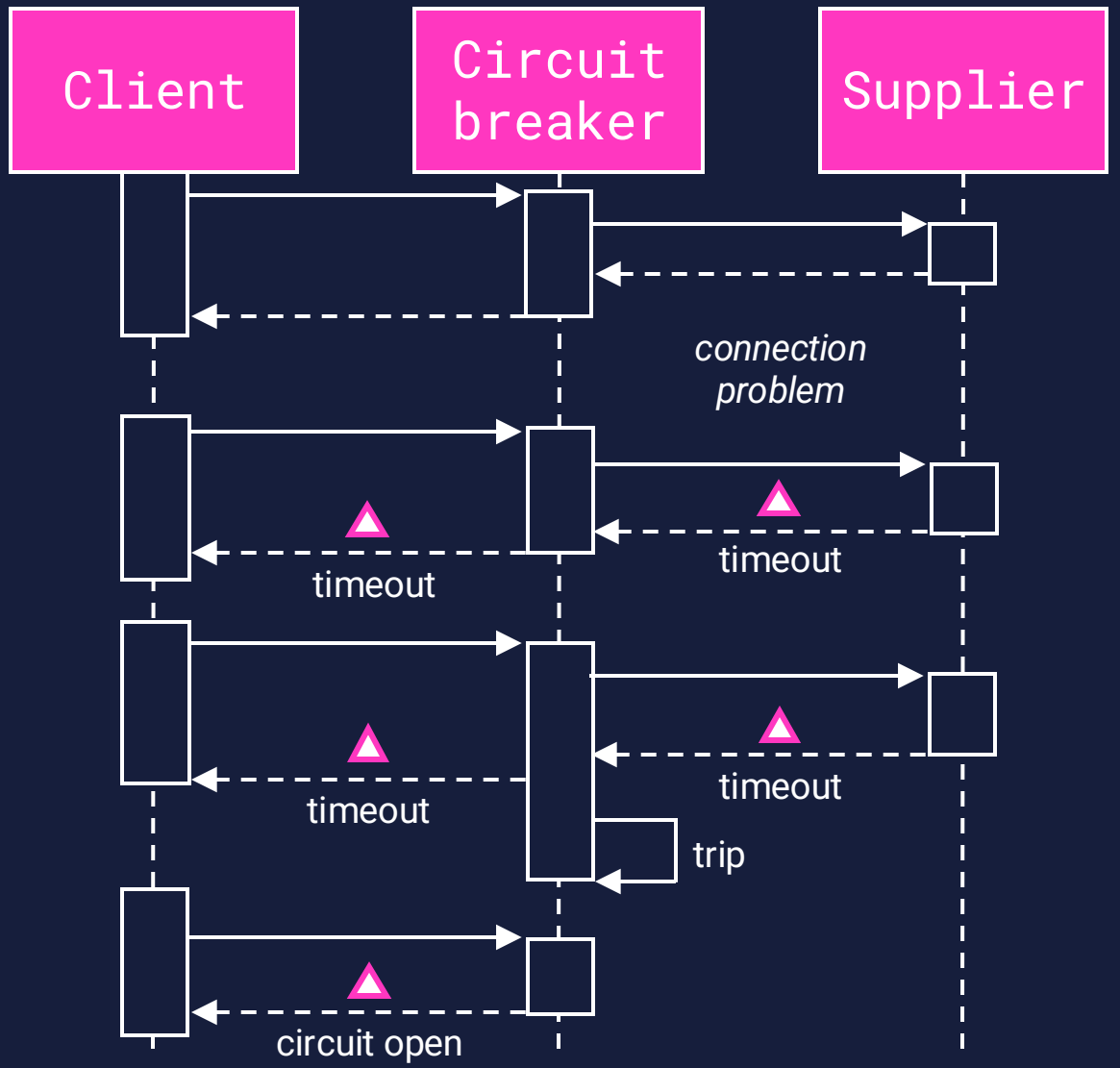
Centralized log analysis

Aggregating logs for better analysis across services

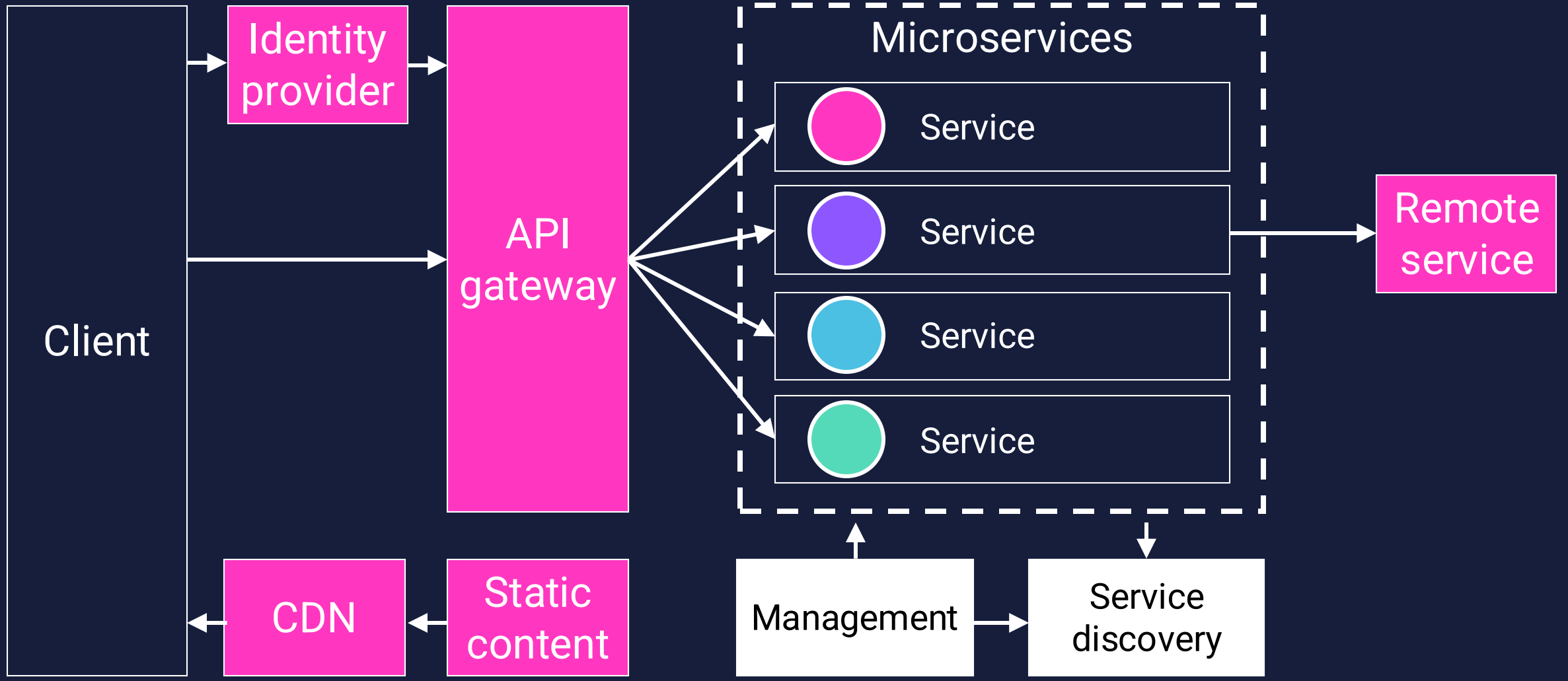
Circuit breaker

Prevent cascading failures with the circuit breaker pattern

Circuit breaker



Architecture view



Exercise

Design a microservices architecture

Microservice design considerations

- Use DDD to identify bounded contexts and services
- Align services with business functions.
- Choose synchronous or asynchronous communication (e.g. REST or gRPC (sync) vs message queues and event driven architecture.
- Database per service or shared database

Challenges

- Network-based inter-process communication
- Handling distributed transactions
- Managing a high volume of services
- Increased need for automation

Which of the following is NOT a core characteristic of microservices architecture?

{A} Each service focuses on a single business capability.

{B} Services can be scaled independently based on demand.

{C} A single database should be shared among all services.

{D} Different services can use different programming languages and technologies.

Which of the following is NOT a core characteristic of microservices architecture?

{A} Each service focuses on a single business capability.

{B} Services can be scaled independently based on demand.

{C} **A single database should be shared among all services.**

{D} Different services can use different programming languages and technologies.

What is the primary advantage of having services communicate asynchronously using message queues?

{A} It ensures services are tightly coupled.

{B} It simplifies the overall system architecture.

{C} It allows services to operate independently, improving fault tolerance.

{D} It eliminates the need for APIs between services.

What is the primary advantage of having services communicate asynchronously using message queues?

{A} It ensures services are tightly coupled.

{B} It simplifies the overall system architecture.

{C} **It allows services to operate independently, improving fault tolerance.**

{D} It eliminates the need for APIs between services.

Which design principle is best described by the idea that each microservice should have one specific responsibility and encapsulate all related functionality?

{A} Open/closed principle

{B} Single responsibility principle

{C} Dependency inversion principle

Which design principle is best described by the idea that each microservice should have one specific responsibility and encapsulate all related functionality?

{A} Open/closed principle

{B} **Single responsibility principle**

{C} Dependency inversion principle

Which strategy is generally recommended for managing data persistence and storage for microservices?

{A} Implement a database per service pattern to allow services to manage their own data independently.

{B} Use a shared database accessible by all microservices to ensure data consistency.

{C} Store all data in local files within each microservice to reduce latency.

{D} Centralize data management in a dedicated data service that handles all database operations for other services.

Which strategy is generally recommended for managing data persistence and storage for microservices?

{A} **Implement a database per service pattern to allow services to manage their own data independently.**

{B} Use a shared database accessible by all microservices to ensure data consistency.

{C} Store all data in local files within each microservice to reduce latency.

{D} Centralize data management in a dedicated data service that handles all database operations for other services.

Next up:

Comparing microservices and monolithic architecture

Questions or suggestions?

maaikejvp@gmail.com

See you tomorrow!