สถาบันไอทีจีเนียส
IT GENIUS INSTITUTE

COURSE ID
WDC-43

COURSE OUTLINE · รายละเอียดหลักสูตรอบรม

WEB DEVELOPMENT / MICROSERVICES

Building Microservices with Node.js

ออกแบบและสร้างระบบ Microservices ด้วย Node.js, Fastify และ RabbitMQ

- 12 ชั่วโมง
- 2 วัน
- พื้นฐาน JavaScript และ Node.js
- ระดับ Intermediate



ระยะเวลา: 2 วัน
12 ชั่วโมง (2 วัน วันละ 6 ชม.) · 09:00-16:00 น.

รูปแบบ: Classroom / Hybrid
Onsite หรือ Online ผ่าน Microsoft Teams · Public / In-house

รายละเอียดหลักสูตร
www.itgenius.co.th

Microservices ช่วยให้แบ่งระบบใหญ่เป็นบริการเล็กหลายตัวที่พัฒนา Deploy และขยายแยกกันได้ ทีมจึงส่งงานได้เร็วขึ้นและขยายเฉพาะส่วนที่มีโหลดสูง Node.js เหมาะกับงานนี้เพราะเบา เริ่มเร็ว และจัดการงาน I/O จำนวนมากได้ดี แต่การแยก Service โดยไม่มีหลักคิดมักได้ระบบที่ซับซ้อนกว่าเดิม ทั้งเรียกกันเป็นลูกโซ่จนล่มตามกัน ข้อมูลไม่ตรงกันระหว่าง Service และหาไม่เจอว่าปัญหาเกิดที่ไหน

หลักสูตรนี้พาสร้างระบบ Microservices ด้วย Node.js แบบไม่พึ่ง Framework ใหญ่ เริ่มจากแบ่ง Service ตามงานธุรกิจ เตรียมโปรเจกต์ Node.js 24 LTS แบบ ES Modules สร้าง Service ด้วย Fastify 5 และ Express 5 ให้แต่ละ Service มีฐานข้อมูลของตัวเอง เรียกกันผ่าน HTTP อย่างทนทาน และสร้าง API Gateway ต่อด้วยการสื่อสารแบบ Event ผ่าน RabbitMQ การรักษาความถูกต้องของข้อมูลข้าม Service การรันทั้งระบบด้วย Docker Compose การ Load Balance ด้วย NGINX และการติดตามระบบด้วย OpenTelemetry แล้วปิดท้ายด้วย Workshop ระบบจองทริปแบบ Microservices (อบรม 2 วัน วันละ 6 ชั่วโมง รวม 12 ชั่วโมง ระดับ Intermediate)

เครื่องมือ

- N Node.js 24 LTS
- F Fastify 5
- Rb RabbitMQ
- Dk Docker Compose
- OT OpenTelemetry

ราคาค่าอบรม · PUBLIC / IN-HOUSE

5,500 บาท / ท่าน

สมัครผ่านเว็บไซต์ ลด 10%

ราคายังไม่รวมภาษีมูลค่าเพิ่ม (VAT 7%) · รวมเอกสารประกอบการอบรม Lab Guide ซอร์สโค้ดตัวอย่างครบทุก Lab และ Certificate · รับจัด In-house ขอใบเสนอราคาได้

เพิ่มเพื่อน LINE

02-570-8449 · 088-807-9770 · @itgenius · www.itgenius.co.th

01 วัตถุประสงค์ Objectives

- เข้าใจสถาปัตยกรรม Microservices ความต่างจาก Monolith และเกณฑ์ตัดสินใจว่าเมื่อไรควรใช้
- แบ่ง Service ตามงานธุรกิจ และออกแบบให้แต่ละ Service เป็นเจ้าของข้อมูลของตัวเอง
- สร้าง Service ด้วย Node.js 24 LTS, Fastify 5 และ Express 5 พร้อม Validation และ OpenAPI
- สร้าง API Gateway และเรียก Service อื่นผ่าน HTTP พร้อม Timeout และ Retry
- สื่อสารแบบ Event ผ่าน RabbitMQ และรับมือกับข้อมูลที่ต้องสอดคล้องกันข้าม Service
- รันทั้งระบบด้วย Docker Compose ขยายระบบด้วย NGINX และติดตามระบบด้วย Log, Health Check และ OpenTelemetry

02 กลุ่มเป้าหมาย Target Audience

- Node.js Developer ที่ต้องการแยกระบบ Monolith เป็น Microservices
- Backend Developer และ Full-stack Developer ที่ใช้ JavaScript หรือ TypeScript
- ผู้ดูแลระบบและทีม DevOps ที่ต้อง Deploy และดูแลระบบหลาย Service
- นักศึกษา ครู อาจารย์ และผู้สนใจที่มีพื้นฐาน Node.js และอยากเข้าใจ Microservices จากการลงมือทำ

03 จุดเด่นของหลักสูตร Highlights

- **Node.js ล้วน ไม่พึ่ง Framework ใหญ่** Fastify 5 และ Express 5
- **สื่อสารได้ทั้งสองแบบ** HTTP และ Event ผ่าน RabbitMQ
- **ทนทานเมื่อมีจุดล่ม** Timeout, Retry และ Graceful Shutdown
- **มองเห็นทั้งระบบ** Log, Health Check และ OpenTelemetry
- **รันได้จริงบนเครื่อง** Docker Compose และ NGINX

04 ความรู้พื้นฐาน Prerequisites

- เขียน JavaScript ได้ ทั้ง Function, Object, Array และ async กับ await
- เคยสร้าง REST API ด้วย Node.js มาบ้าง
- ใช้งานฐานข้อมูลและ Command Line ขั้นพื้นฐานได้
- โน้ตบุ๊กที่ติดตั้ง Node.js รุ่น LTS, Docker Desktop และ VS Code ได้

05 สิ่งที่ต้องเตรียมและเครื่องมือ Setup & Tools

รายการ	รายละเอียด
อุปกรณ์	Notebook Windows, macOS หรือ Linux เชื่อมต่ออินเทอร์เน็ต พร้อมสิทธิ์ติดตั้งโปรแกรม
Runtime และ Framework	Node.js 24 LTS, npm Workspaces, Fastify 5, Express 5 และ VS Code
ข้อมูลและ Messaging	MongoDB และ RabbitMQ บน Docker พร้อม amqplib
Infrastructure	Docker Desktop, Docker Compose, NGINX, OpenTelemetry และ Jaeger

หมายเหตุ: หลักสูตรนี้ปรับปรุงใหม่ปี 2026 จากฉบับเดิมที่ใช้ Express รุ่นเก่ากับ Callback และ Promise, สร้าง Test Harness ด้วย React, Deploy ด้วย PM2 และมอนิเตอร์ผ่านบริการภายนอก ฉบับนี้ใช้ Node.js 24 LTS แบบ ES Modules, Fastify 5 เป็นหลักคู่กับ Express 5 ที่ส่ง Error จาก async ไปยัง Error Middleware อัตโนมัติ เปลี่ยนการ Deploy หลักเป็น Docker Compose เพิ่มการสื่อสารแบบ Event ผ่าน RabbitMQ และติดตามระบบด้วย OpenTelemetry ส่วน NGINX สำหรับ Load Balance และ API Gateway ยังคงอยู่ หลักสูตรนี้ใช้ Node.js ล้วน ต่างจาก Building Microservices with NestJS (WDC-92) ที่ใช้ NestJS Framework เครื่องมือทั้งหมดเป็นโอเพนซอร์สและรันบนเครื่องของผู้เรียน

06 ภาพรวมการเรียน 2 วัน Big Picture

วัน	หัวข้อหลัก
Day 1 Section 1-6	แบ่งระบบเป็น Service ตามงานธุรกิจ, เตรียมโปรเจกต์ Node.js 24 LTS แบบ ES Modules, สร้าง Service ด้วย Fastify 5 และ Express 5, ให้แต่ละ Service มีฐานข้อมูลของตัวเอง เรียกกันผ่าน HTTP อย่างทนทาน และสร้าง API Gateway
Day 2 Section 7-12	สื่อสารแบบ Event ผ่าน RabbitMQ, รักษาความถูกต้องของข้อมูลข้าม Service, รันทั้งระบบด้วย Docker Compose, ขยายระบบและ Load Balance ด้วย NGINX, ติดตามระบบด้วย Log, Health Check และ OpenTelemetry

DAY 1

ออกแบบและสร้าง Service

เนื้อหาการอบรม · Section 1-6

รวม
6 ชั่วโมง

1 Microservices คืออะไร และเมื่อไรควรใช้

- Monolith, Modular Monolith และ Microservices ต่างกันอย่างไร
- แบ่ง Service ตามงานธุรกิจด้วยแนวคิด Bounded Context
- ต้นทุนที่ต้องจ่าย: เครือข่าย ข้อมูล การ Deploy และการติดตามระบบ
- Lab: แบ่งระบบจองทริปท่องเที่ยวเป็น Service และวาดภาพสถาปัตยกรรม

2 Lab: เตรียมโปรเจกต์ Node.js 24 LTS

- Monorepo ด้วย npm Workspaces เทียบกับ Multi-repo
- ES Modules, async กับ await และ EventEmitter ที่ใช้ใน Service
- node --watch, --env-file และ Test Runner ในตัว (node:test)
- Lab: โครงโปรเจกต์ที่มีหลาย Service และไลบรารีใช้ร่วมกัน

3 Lab: Service แรกด้วย Fastify 5

- Route, Plugin และ Validation ด้วย JSON Schema
- Logging ด้วย Pino และสร้างเอกสาร OpenAPI อัตโนมัติ
- เทียบกับ Express 5: Router, Middleware และการจัดการ Error ของ async
- Lab: Catalog Service ที่มี CRUD API พร้อม Schema และเอกสาร

4 Lab: ข้อมูลของแต่ละ Service

- Database per Service และเหตุผลที่ไม่ควรใช้ฐานข้อมูลร่วมกัน
- เชื่อม MongoDB บน Docker และออกแบบข้อมูลของแต่ละ Service
- API Composition เมื่อหน้าจอต้องใช้ข้อมูลจากหลาย Service
- Lab: Booking Service ที่มีฐานข้อมูลแยกของตัวเอง

5 Lab: เรียก Service อื่นผ่าน HTTP อย่างทนทาน

- เรียก Service ด้วย fetch ในตัวของ Node.js พร้อม AbortSignal.timeout
- Retry แบบ Backoff, Idempotency Key และการป้องกันการเรียกซ้ำ
- แนวคิด Circuit Breaker และ Fallback เมื่อปลายทางล่ม
- Lab: Booking เรียก Catalog เพื่อตรวจที่ว่างและรับมือเมื่อ Catalog ล่ม

6 Lab: API Gateway

- หน้าที่ของ API Gateway: Routing, ยืนยันตัวตน และจำกัดจำนวน Request
- สร้าง Gateway ด้วย Fastify, @fastify/http-proxy และ @fastify/rate-limit
- ตรวจ JWT ที่ Gateway และส่งข้อมูลผู้ใช้ต่อให้ Service
- Lab: รวมทุก Service ไว้หลัง Gateway เดียว

DAY 2

Messaging, Deploy และการติดตามระบบ

เนื้อหาการอบรม · Section 7-12

รวม
6 ชั่วโมง

7 Lab: สื่อสารแบบ Event ผ่าน RabbitMQ

- Synchronous เทียบกับ Asynchronous และเมื่อไรควรใช้ Event
- Exchange, Queue, Routing Key และ Durable Queue
- เขียน Publisher และ Consumer ด้วย amqplib พร้อม Ack และ Prefetch
- Lab: Booking ส่ง Event ให้ Notification Service ส่งอีเมลยืนยัน

8 ความถูกต้องของข้อมูลข้าม Service

- Eventual Consistency และสิ่งที่ผู้ใช้จะเห็นระหว่างทาง
- Saga แบบ Choreography และการชดเชยเมื่อขั้นตอนใดล้มเหลว
- Outbox Pattern และ Consumer ที่รับข้อความซ้ำได้อย่างปลอดภัย
- Lab: ยกเลิกการจองอัตโนมัติเมื่อชำระเงินไม่สำเร็จ

9 Lab: รันทั้งระบบด้วย Docker Compose

- Dockerfile แบบ Multi-stage สำหรับ Node.js และการรันแบบ Non-root
- Docker Compose สำหรับทุก Service, MongoDB และ RabbitMQ
- Environment Variable, Healthcheck และลำดับการเริ่มระบบ
- เทียบ PM2 กับ Container และภาพรวมการขึ้น Kubernetes

10 Lab: ขยายระบบและ Load Balancing ด้วย NGINX

- ออกแบบ Service ให้ Stateless เพื่อเพิ่มจำนวน Instance ได้
- NGINX upstream แบบ Round Robin และ least_conn
- Graceful Shutdown เมื่อได้รับ SIGTERM
- Lab: เพิ่ม Instance ของ Catalog แล้วทดสอบด้วยคำขอจำนวนมาก

11 Lab: Log, Health Check และ Tracing

- Structured Log ด้วย Pino และ Correlation ID ข้าม Service
- Health และ Readiness Endpoint สำหรับ Docker และ Load Balancer
- Distributed Tracing ด้วย OpenTelemetry แล้วดูผลใน Jaeger
- Lab: ไล่หา Service ที่ทำให้ Request ช้าจาก Trace

12 Workshop: ระบบจองทริปแบบ Microservices

- ประกอบ Catalog, Booking, Payment และ Notification Service เข้าด้วยกัน
- เขียน Test ของ Service ด้วย node:test และทดสอบทั้งระบบบน Docker Compose
- ทดลองปิดบาง Service แล้วดูว่าระบบรับมืออย่างไร
- นำเสนอผลงาน รีวิวร่วมกัน และสรุป Checklist ก่อนขึ้น Production

07 เรียนจบแล้วได้อะไร Outcomes

- **ออกแบบ Service ได้** แบ่งตามงานธุรกิจและข้อมูล
- **สร้าง Service ได้** Node.js 24, Fastify 5 และ Express 5
- **เชื่อม Service ได้** HTTP, API Gateway และ RabbitMQ
- **ระบบทนทานขึ้น** Timeout, Retry, Saga และ Outbox
- **Deploy ได้เอง** Docker Compose และ NGINX
- **ติดตามระบบได้** Log, Health Check และ Tracing

08 สิ่งที่ผู้เรียนจะได้รับ Deliverables

- เอกสารประกอบการอบรมครบทั้ง 2 วัน และ Lab Guide ทีละขั้นตอน
- ซอร์สโค้ดตัวอย่างครบทุก Lab และโปรเจกต์ระบบจองทริปแบบ Microservices
- ไฟล์ Docker Compose และ NGINX Config สำหรับรันทั้งระบบบนเครื่อง
- ประกาศนียบัตรจบหลักสูตร (Certificate of Completion) และช่องทางสอบถามวิทยากรหลังจบหลักสูตร

09 ข้อมูลผู้สอน Instructor

อ.สนิทวงศ์ กมลภากรณ์ ผู้สอนและนักพัฒนาที่ใช้ Node.js สร้างระบบหลังบ้านและแยกระบบเป็นหลาย Service ในงานจริง สอนผ่านระบบตัวอย่างชุดเดียวที่ค่อย ๆ โตขึ้นตลอดหลักสูตร ให้ผู้เรียนเห็นปัญหาที่เกิดขึ้นจริงเมื่อ Service เรียกกันหรือล่มบางส่วน แล้วแก้ทีละจุดด้วยหลักการที่เหมาะสม เพื่อให้กลับไปตัดสินใจได้เองว่าระบบของตนควรแยกแค่ไหนและควรสื่อสารกันแบบใด

ราคาค่าอบรม 5,500 บาท/ท่าน · สนใจ IN-HOUSE ขอใบเสนอราคา

5,500 บาท / ท่าน (ยังไม่รวม VAT 7%)

สมัครผ่านเว็บไซต์ ลด 10%

02-570-8449 · 088-807-9770

@itgenius · contact@itgenius.co.th

เพิ่มเพื่อน LINE