สถาบันไอทีจีเนียส
IT GENIUS INSTITUTE

COURSE ID
**WDC-57**
WEB DEVELOPMENT / .NET

COURSE OUTLINE · รายละเอียดหลักสูตรอบรม

# Building Microservices with .NET

## ออกแบบและสร้างระบบ Microservices ด้วย .NET 10 และ Aspire

- 18 ชั่วโมง
- 3 วัน
- พื้นฐาน C# และ ASP.NET Core Web API
- ระดับ Intermediate




**ระยะเวลา: 3 วัน**
18 ชั่วโมง (3 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 แบ่งระบบใหญ่ออกเป็นเซอร์วิสย่อยที่พัฒนา Deploy และขยายได้อย่างอิสระ แต่ละเซอร์วิสมีฐานข้อมูลของตัวเองและคุยกันผ่าน API หรือ Message ข้อดีคือทีมทำงานคู่ขนานได้และเปลี่ยนแปลงทีละส่วนได้ แต่ก็ตามมาด้วยความซับซ้อนของระบบกระจาย ทั้งเซอร์วิสที่เรียกกันแล้วล้มต่อกันเป็นทอด ๆ ข้อมูลที่ไม่ตรงกันข้ามเซอร์วิส และการไล่หาปัญหาที่เกิดขึ้นระหว่างหลายเซอร์วิส หลายทีมจึงแยกระบบเร็วเกินไปและได้ระบบที่ดูแลยากกว่า Monolith เดิม

หลักสูตรนี้พาผู้เรียนออกแบบและสร้างระบบ Microservices ด้วย .NET 10 และ ASP.NET Core 10 ตั้งแต่ตัดสินใจว่าเมื่อไรควรแยกระบบ แบ่งขอบเขตเซอร์วิสด้วยแนวคิด DDD สร้างเซอร์วิสที่มีฐานข้อมูลของตัวเอง แล้วประกอบและรันทั้งระบบด้วย Aspire จากนั้นฝึกการสื่อสารแบบ HTTP, gRPC และ Message ผ่าน RabbitMQ เพิ่ม Resilience จัดการ Transaction ข้ามเซอร์วิสด้วย Saga และ Outbox วาง API Gateway ด้วย YARP ทำ Event Sourcing และ CQRS ดูแลความปลอดภัยและ Observability ด้วย OpenTelemetry ทดสอบด้วย xUnit และ Testcontainers ปิดท้ายด้วยการ Build เป็น Container และ Capstone ระบบร้านค้าแบบ Microservices (อบรม 3 วัน วันละ 6 ชั่วโมง รวม 18 ชั่วโมง ระดับ Intermediate)

เครื่องมือ: .NET 10 · Aspire 13 · RabbitMQ · PostgreSQL · Docker

ราคาค่าอบรม · PUBLIC / IN-HOUSE

**10,000** บาท / ท่าน

สมัครผ่านเว็บไซต์ ลด 10%

ราคายังไม่รวมภาษีมูลค่าเพิ่ม (VAT 7%) · รวมเอกสารประกอบการอบรม Lab Guide ซอร์สโค้ดระบบตัวอย่างครบทุก Lab ไฟล์ docker-compose พร้อมใช้ และ Certificate · รับจัด In-house ขอใบเสนอราคาได้

เพิ่มเพื่อน LINE

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

## 01 วัตถุประสงค์ Objectives

- ตัดสินใจได้ว่าเมื่อไรควรใช้ Microservices และเมื่อไร Modular Monolith เหมาะกว่า
- แบ่งขอบเขตเซอร์วิสด้วย DDD และออกแบบฐานข้อมูลแยกตามเซอร์วิส
- สร้างเซอร์วิสด้วย ASP.NET Core 10 และรันทั้งระบบด้วย Aspire
- สื่อสารระหว่างเซอร์วิสด้วย HTTP, gRPC และ Message ผ่าน RabbitMQ พร้อม Resilience
- จัดการ Transaction ข้ามเซอร์วิสด้วย Saga และ Outbox Pattern
- ประยุกต์ Event Sourcing, CQRS และ API Gateway ได้อย่างเหมาะสม
- ติดตามระบบด้วย OpenTelemetry และทดสอบ Microservices ได้หลายระดับ
- Build เซอร์วิสเป็น Container และเตรียม Deploy ด้วย Docker Compose

## 02 กลุ่มเป้าหมาย Target Audience

- นักพัฒนา .NET และ C# ที่ต้องการออกแบบระบบแบบ Microservices
- นักพัฒนาที่ดูแลระบบ Monolith ขนาดใหญ่และกำลังพิจารณาแยกระบบ
- Software Architect และ Tech Lead ที่ต้องตัดสินใจเรื่องสถาปัตยกรรมของระบบ
- ทีม DevOps ที่ต้องรองรับการ Deploy และติดตามระบบที่มีหลายเซอร์วิส
- ครู อาจารย์ และนักศึกษาสายคอมพิวเตอร์ที่สนใจสถาปัตยกรรมซอฟต์แวร์สมัยใหม่

## 03 จุดเด่นของหลักสูตร Highlights

- **อัปเดตเป็น .NET 10** ASP.NET Core 10, EF Core 10 และ Aspire 13
- **ออกแบบก่อนแยก** Modular Monolith, DDD และ Database per Service
- **สื่อสารอย่างทนทาน** HTTP, gRPC, RabbitMQ และ Resilience
- **ข้อมูลที่สอดคล้อง** Saga, Outbox, Event Sourcing และ CQRS
- **พร้อมขึ้นระบบจริง** OpenTelemetry, Testing และ Container

## 04 ความรู้พื้นฐาน Prerequisites

- เขียนภาษา C# ได้ดี และเคยสร้าง Web API ด้วย ASP.NET Core มาก่อน
- รู้จักฐานข้อมูลเชิงสัมพันธ์ คำสั่ง SQL และ Entity Framework Core เบื้องต้น
- เข้าใจ HTTP, JSON และหลักการของ RESTful API
- เคยใช้ Docker และ Git ขั้นพื้นฐาน
- โน้ตบุ๊ก RAM 16 GB ขึ้นไป ที่ติดตั้ง .NET 10 SDK และ Docker Desktop ได้ พร้อมสิทธิ์ติดตั้งโปรแกรม

## 05 สิ่งที่ต้องเตรียมและเครื่องมือ Setup & Tools

| รายการ | รายละเอียด |
|---|---|
| **อุปกรณ์** | Notebook RAM 16 GB ขึ้นไป เชื่อมต่ออินเทอร์เน็ต ติดตั้ง Docker Desktop ได้ |
| **แพลตฟอร์ม** | .NET 10 SDK (LTS), C# 14, ASP.NET Core 10, EF Core 10 และ Aspire CLI |
| **เครื่องมือพัฒนา** | Visual Studio 2026 Community หรือ VS Code พร้อม C# Dev Kit |
| **โครงสร้างพื้นฐาน** | PostgreSQL, RabbitMQ, Redis และ YARP บน Docker ผ่าน Aspire AppHost |
| **ทดสอบและติดตามระบบ** | xUnit, Testcontainers for .NET, Aspire Dashboard และ OpenTelemetry |

**หมายเหตุ:** หลักสูตรนี้ปรับปรุงใหม่ปี 2026 จากฉบับเดิมที่สอนบน .NET Core เนื้อหาเปลี่ยนเป็น .NET 10 (LTS) ทั้งหมด ใช้ Aspire ประกอบและรันระบบแทนการเขียน docker-compose ทีละเซอร์วิส เพิ่ม gRPC, Resilience, RabbitMQ, Saga และ Outbox, YARP, OpenTelemetry และ Testcontainers โดยคงหัวข้อเดิมเรื่องการแยกระบบ Transaction, Event Sourcing, Querying และการทดสอบไว้ในรูปแบบปัจจุบัน Lab ส่งข้อความด้วย RabbitMQ.Client โดยตรง เพราะ MassTransit ตั้งแต่รุ่น 9 เป็นซอฟต์แวร์เชิงพาณิชย์ที่ต้องซื้อสิทธิ์ใช้งาน เครื่องมือที่ใช้ใน Lab ทั้งหมดใช้ได้ฟรี Visual Studio Community มีเงื่อนไขสิทธิ์การใช้งานสำหรับองค์กร หากไม่เข้าเงื่อนไขใช้ VS Code แทนได้ทุก Lab

## 06 ภาพรวมการเรียน 3 วัน Big Picture

| วัน | หัวข้อหลัก |
|---|---|
| **Day 1**<br>Section 1-5 | เตรียม .NET 10 และ Aspire, เทียบ Monolith, Modular Monolith และ Microservices, แยกเซอร์วิสด้วย DDD, สร้างเซอร์วิสด้วย ASP.NET Core 10 แล้วประกอบทั้งระบบด้วย Aspire AppHost |
| **Day 2**<br>Section 6-10 | สื่อสารระหว่างเซอร์วิสด้วย HTTP และ gRPC, เพิ่ม Resilience, ส่ง Event ผ่าน RabbitMQ, จัดการ Transaction ข้ามเซอร์วิสด้วย Saga และ Outbox แล้ววาง API Gateway ด้วย YARP |
| **Day 3**<br>Section 11-15 | ทำ Event Sourcing และ CQRS, Query ข้ามเซอร์วิส, ความปลอดภัยและ Observability ด้วย OpenTelemetry, ทดสอบด้วย xUnit และ Testcontainers แล้วปิดท้ายด้วย Container, Deploy และ Capstone |

DAY 1

# ออกแบบและสร้างเซอร์วิสแรก

เนื้อหาการอบรม · Section 1-5

รวม
**6 ชั่วโมง**

## 1 Lab: เตรียม .NET 10 และ Aspire

- ภาพรวม .NET 10 (LTS), C# 14 และ ASP.NET Core 10
- ติดตั้ง .NET 10 SDK, Docker Desktop และ Aspire CLI
- สร้างโซลูชันด้วย aspire new และรู้จัก AppHost กับ ServiceDefaults
- สำรวจ Aspire Dashboard: Resource, Log, Trace และ Metric
- Lab: รันระบบตัวอย่างแรกด้วย aspire run

## 2 Monolith, Modular Monolith และ Microservices

- ข้อดีและต้นทุนของ Microservices ที่มักถูกมองข้าม
- Modular Monolith เป็นจุดเริ่มต้นที่ปลอดภัยกว่าสำหรับหลายทีม
- สัญญาณที่บอกว่าถึงเวลาแยกเซอร์วิส และสัญญาณที่บอกว่ายังไม่ควร
- ความท้าทายของระบบกระจาย: Network, Latency และ Consistency

## 3 แยกระบบเป็นเซอร์วิสด้วย DDD

- Bounded Context และ Ubiquitous Language
- Aggregate และการวาง Business Logic ให้อยู่ในเซอร์วิสที่ถูกต้อง
- Database per Service และเหตุผลที่ไม่ควรแชร์ฐานข้อมูล
- Workshop: แบ่งระบบร้านค้าเป็นเซอร์วิส Catalog, Ordering และ Payment

## 4 Lab: สร้างเซอร์วิสด้วย ASP.NET Core 10

- สร้าง Catalog Service ด้วย Minimal API และเอกสาร OpenAPI ในตัว
- จัดการข้อมูลด้วย EF Core 10 กับ PostgreSQL และ Migration
- Validation, Problem Details และรูปแบบ Error มาตรฐาน
- Health Check และ Configuration ของแต่ละเซอร์วิส
- Lab: สร้าง Catalog และ Ordering Service ที่มีฐานข้อมูลแยกกัน

## 5 Lab: ประกอบทั้งระบบด้วย Aspire

- ประกาศ Resource ใน AppHost ด้วย AddProject และ AddPostgres
- ส่ง Connection String และ Endpoint ด้วย WithReference
- Service Discovery ระหว่างเซอร์วิสโดยไม่ต้องกำหนด URL เอง
- Hosting Integration และ Client Integration ของ Aspire
- Lab: รันทุกเซอร์วิสพร้อมฐานข้อมูลด้วยคำสั่งเดียว

DAY 2

# การสื่อสารระหว่างเซอร์วิส

เนื้อหาการอบรม · Section 6-10

รวม
**6 ชั่วโมง**

## 6 การสื่อสารแบบ Synchronous

- เรียกเซอร์วิสอื่นด้วย Typed HttpClient และ Service Discovery
- gRPC สำหรับการสื่อสารภายในที่ต้องการความเร็ว
- เลือกระหว่าง REST และ gRPC และการทำ Versioning ของสัญญา
- Lab: Ordering Service ตรวจราคาสินค้าจาก Catalog ผ่าน gRPC

## 7 Lab: Resilience และการเชื่อมต่อ API ภายนอก

- Timeout, Retry และ Circuit Breaker ด้วย Microsoft.Extensions.Http.Resilience
- Standard Resilience Handler และการปรับค่าให้เหมาะกับแต่ละปลายทาง
- Idempotency เพื่อให้เรียกซ้ำได้อย่างปลอดภัย
- เชื่อมต่อ API ภายนอก เช่น Payment Gateway จำลอง
- Lab: ทดลองปิดเซอร์วิสปลายทางแล้วดูพฤติกรรมของระบบ

## 8 Lab: Asynchronous Messaging ด้วย RabbitMQ

- Command เทียบกับ Event และรูปแบบ Publish/Subscribe
- เพิ่ม RabbitMQ ใน AppHost และเชื่อมด้วย RabbitMQ.Client รุ่น 7 แบบ Async
- Exchange, Queue, Acknowledgement และ Dead Letter Queue
- ภาพรวมไลบรารี Messaging อื่นและเงื่อนไขสิทธิ์การใช้งาน
- Lab: ส่ง Event OrderCreated ไปให้ Payment Service ประมวลผล

## 9 Transaction ข้ามเซอร์วิส

- เหตุผลที่ไม่ใช้ Distributed Transaction และแนวคิด Eventual Consistency
- Saga แบบ Choreography และ Orchestration พร้อม Compensation
- Outbox Pattern เพื่อบันทึกข้อมูลและส่ง Event ให้สอดคล้องกัน
- Lab: Saga สั่งซื้อที่ยกเลิกคำสั่งเมื่อชำระเงินไม่สำเร็จ

## 10 Lab: API Gateway ด้วย YARP

- บทบาทของ API Gateway และรูปแบบ Backend for Frontend
- เพิ่ม YARP ใน AppHost และกำหนด Route ไปยังแต่ละเซอร์วิส
- ตรวจสอบสิทธิ์และจำกัดจำนวนคำขอที่ Gateway
- Lab: เปิดระบบร้านค้าให้ Client เรียกผ่าน Gateway จุดเดียว

DAY 3

## ข้อมูล คุณภาพ และการนำขึ้นระบบ

เนื้อหาการอบรม · Section 11-15

รวม 6 ชั่วโมง

### 11 Event Sourcing และ CQRS

- แนวคิด Event Sourcing: เก็บเหตุการณ์แทนสถานะล่าสุด
- Event Store, Projection และการสร้าง Read Model
- CQRS แยกฝั่งเขียนและฝั่งอ่าน และเมื่อไรจึงคุ้มค่า
- Lab: เก็บประวัติคำสั่งซื้อแบบ Event Sourcing บน PostgreSQL

### 12 Querying ข้ามเซอร์วิส

- API Composition สำหรับรวมข้อมูลจากหลายเซอร์วิส
- สร้าง Read Model จาก Event เพื่อลดการเรียกข้ามเซอร์วิส
- Distributed Cache ด้วย Redis ผ่าน Aspire
- Lab: หน้าสรุปคำสั่งซื้อที่รวมข้อมูลจาก Catalog และ Ordering

### 13 ความปลอดภัยและ Observability

- ยืนยันตัวตนด้วย JWT Bearer จาก Identity Provider เช่น Keycloak
- การยืนยันตัวตนระหว่างเซอร์วิสและการจัดการ Secret
- Log, Metric และ Distributed Tracing ด้วย OpenTelemetry
- ไล่หาสาเหตุปัญหาข้ามเซอร์วิสจาก Trace ใน Aspire Dashboard

### 14 Lab: ทดสอบ Microservices

- Unit Test ของ Business Logic ด้วย xUnit
- Integration Test ด้วย WebApplicationFactory และ Testcontainers for .NET
- ทดสอบทั้งระบบด้วย Aspire.Hosting.Testing
- แนวคิด Contract Testing เพื่อป้องกันการเปลี่ยน API ที่กระทบเซอร์วิสอื่น

### 15 Workshop: Container, Deploy และ Capstone

- Build Container Image ด้วย dotnet publish โดยไม่ต้องเขียน Dockerfile
- สร้างไฟล์ Docker Compose จาก AppHost ด้วย aspire publish
- ภาพรวมการ Deploy ขึ้น Kubernetes หรือ Cloud และ CI/CD
- Capstone: เพิ่มเซอร์วิสใหม่เข้าระบบร้านค้าพร้อม Event, Test และ Trace
- นำเสนอผลงานและรีวิวการออกแบบร่วมกัน

## 07 เรียนจบแล้วได้อะไร Outcomes

- **ตัดสินใจเชิงสถาปัตยกรรมได้** Monolith, Modular Monolith หรือ Microservices
- **แบ่งเซอร์วิสได้ถูกขอบเขต** DDD และ Database per Service
- **สร้างและรันระบบได้** ASP.NET Core 10 และ Aspire
- **สื่อสารอย่างทนทาน** gRPC, RabbitMQ และ Resilience
- **รักษาความสอดคล้องของข้อมูล** Saga, Outbox และ Event Sourcing
- **ดูแลระบบได้จริง** OpenTelemetry, Testing และ Container

## 08 สิ่งที่ผู้เรียนจะได้รับ Deliverables

- เอกสารประกอบการอบรมฉบับสมบูรณ์ครบทั้ง 3 วัน และ Lab Guide ทีละขั้นตอน
- ซอร์สโค้ดระบบร้านค้าแบบ Microservices และ Aspire AppHost ครบทุก Lab
- ไฟล์ docker-compose ที่สร้างจาก Aspire พร้อมตัวอย่างการตั้งค่า Gateway และ Messaging
- ประกาศนียบัตรจบหลักสูตร (Certificate of Completion) และช่องทางสอบถามวิทยากรหลังจบหลักสูตร

## 09 ข้อมูลผู้สอน Instructor

**อ.สนิทวงศ์ กมลภากรณ์** ผู้สอนและนักพัฒนาระบบที่ออกแบบและพัฒนาระบบด้วย .NET และ ASP.NET Core ในงานจริง จุดเด่นคือการสอนสถาปัตยกรรมผ่านระบบตัวอย่างชุดเดียวที่ค่อย ๆ แยกเป็นเซอร์วิส ชี้ให้เห็นทั้งข้อดีและต้นทุนของแต่ละแนวทาง และให้ผู้เรียนลองทำให้ระบบล้มแล้วแก้ด้วยตนเอง เพื่อกลับไปตัดสินใจเรื่องสถาปัตยกรรมของระบบตนเองได้อย่างมีเหตุผล

ราคาค่าอบรม 10,000 บาท/ท่าน · สนใจ IN-HOUSE ขอใบเสนอราคา

**10,000 บาท** / ท่าน (ยังไม่รวม VAT 7%)

สมัครผ่านเว็บไซต์ ลด 10%

02-570-8449 · 088-807-9770

@itgenius · contact@itgenius.co.th

เพิ่มเพื่อน LINE