.NET Conf 2022

November 8 – 11, 2022

ORGANIZER DEV CAFE Connect and share SPONSOR Nash Tech.

.NET Conf 2022 Vietnam

19 Nov 2022

Building Microservices in .NET 7

Thang Chung

Sr Solution Architect @NashTech

Microsoft Azure MVP

https://github.com/thangchung

.NET at Microsoft - Upgrading to .NET 6

Microsoft Graph

70B requests / day
⇩37% CPU
⇩91% Operational costs

Azure Active Directory

185B requests / day
⇩67% CPU
⇧50% app efficiency

Microsoft Teams

50 projects, 700+ APIs
⇩25% CPU
⇩60% P99 latency

Azure Cosmos DB

⇩30-65% CPU
⇧5x throughput
⇩85% P95 latency

Exchange Online

⇩26% CPU
⇩80% P99 latency
⇩20% Heap size

Azure App Service

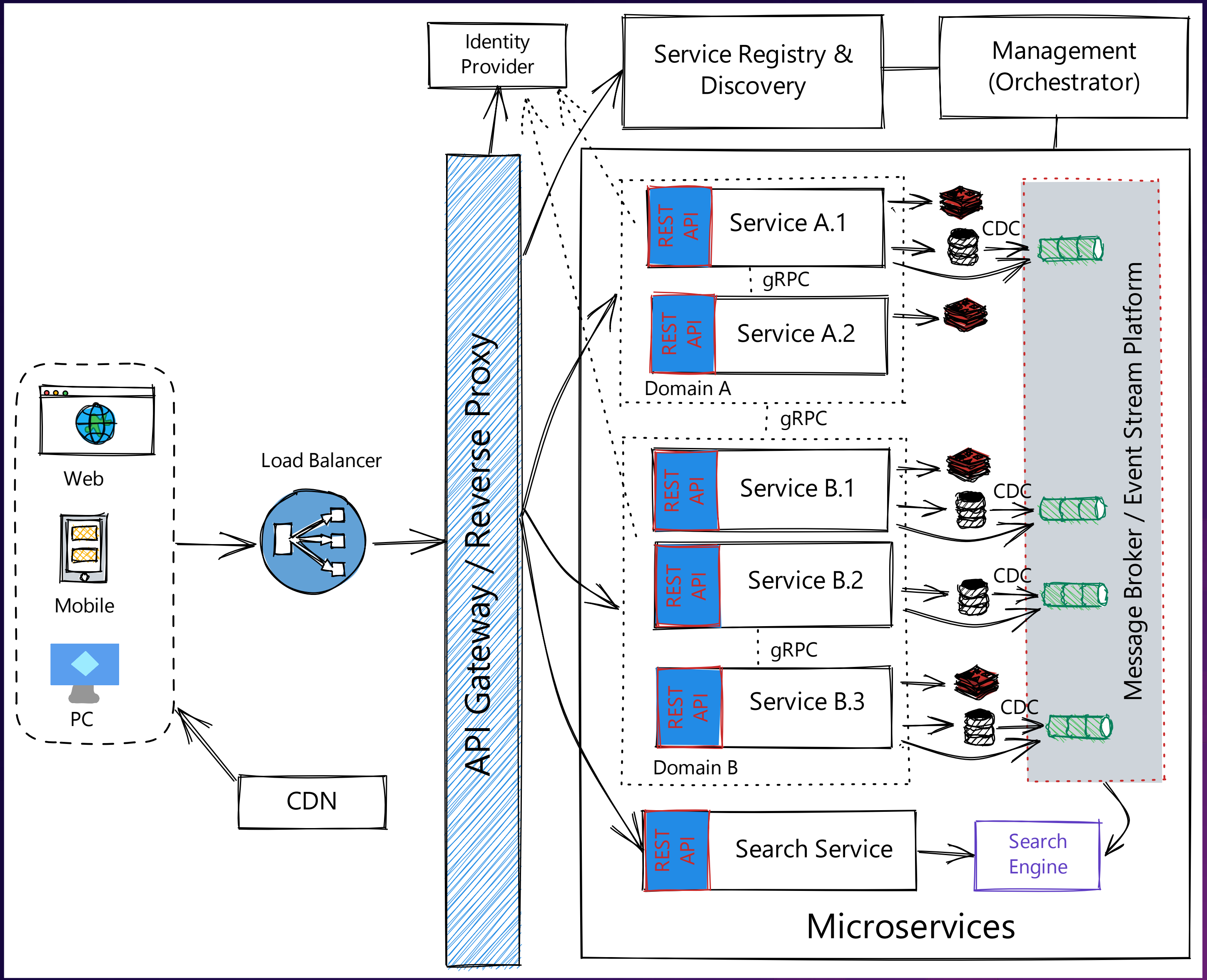
⇧ 80% throughput
Large ⇩ in CPU
Enable gRPC & HTTP/2

Microsoft Graph's Journey to .NET 6
The Azure Cosmos DB Journey to .NET 6

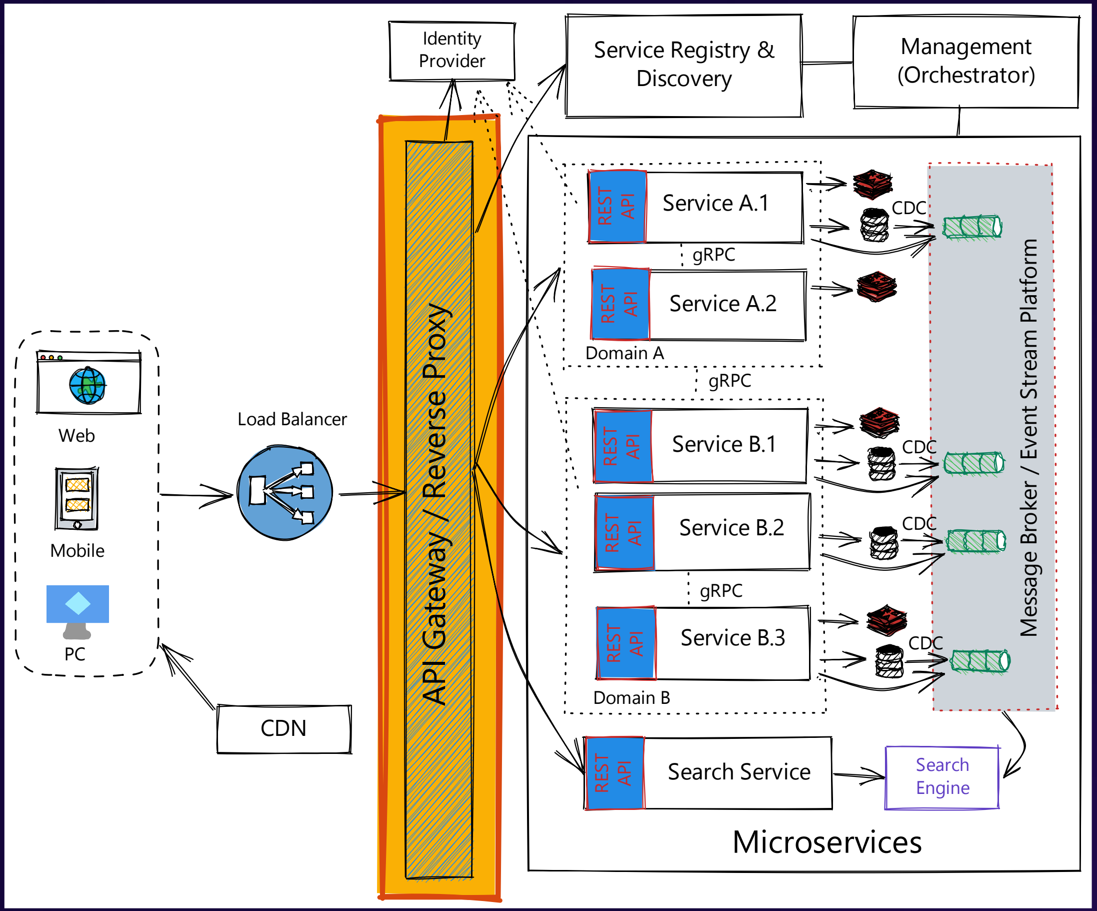
Azure Active Directory's gateway is on .NET 6.0!
Exchange Online Journey to .NET Core

Microsoft Teams' journey to .NET Core
Microsoft Teams Assignments Service's Journey to .NET 6
A Heavy Lift: Bringing Kestrel + YARP to Azure App Services

Popular Microservices High Level Architecture

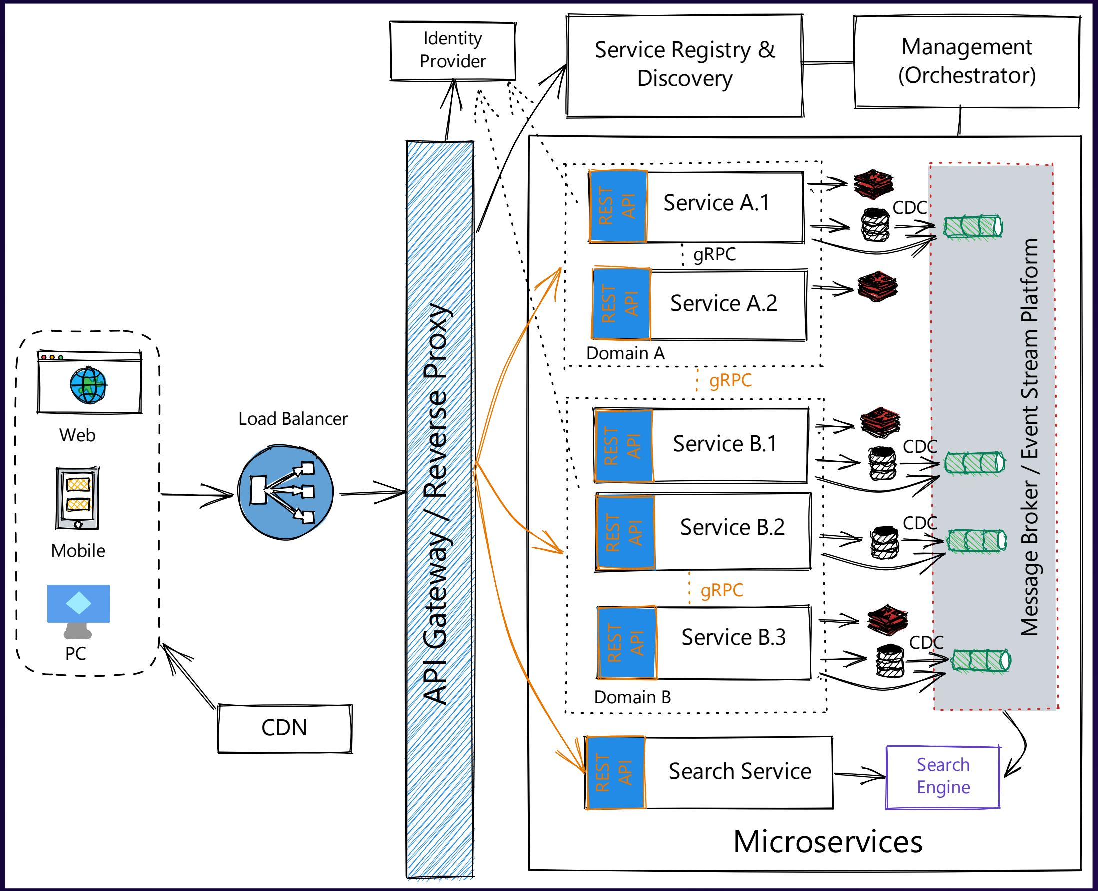


1. API Gateway / Reverse Proxy



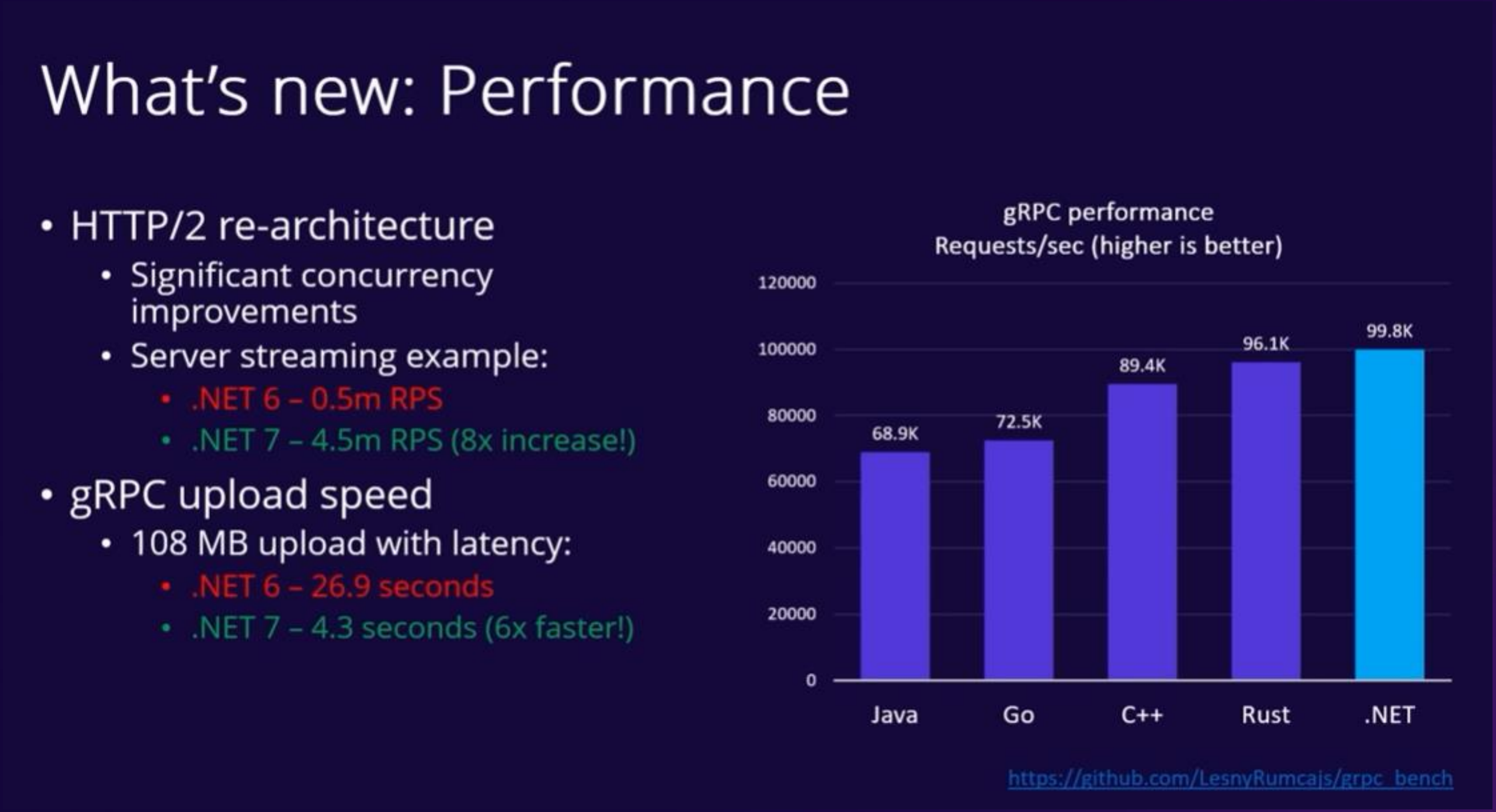
- YARP
  - IHttpForwarder to handle all the boilerplate logic to proxy an incoming HTTP request to another backend
  - This handles HTTP (1, 2 and 3), websockets and gRPC proxying out of the box
- Problem Details (RFC 7807)

2. Synchronous / Asynchronous Communication (1/2)



- RateLimiter (DEMO)
- gRPC

2. Synchronous / Asynchronous Communication – gRPC (2/2)

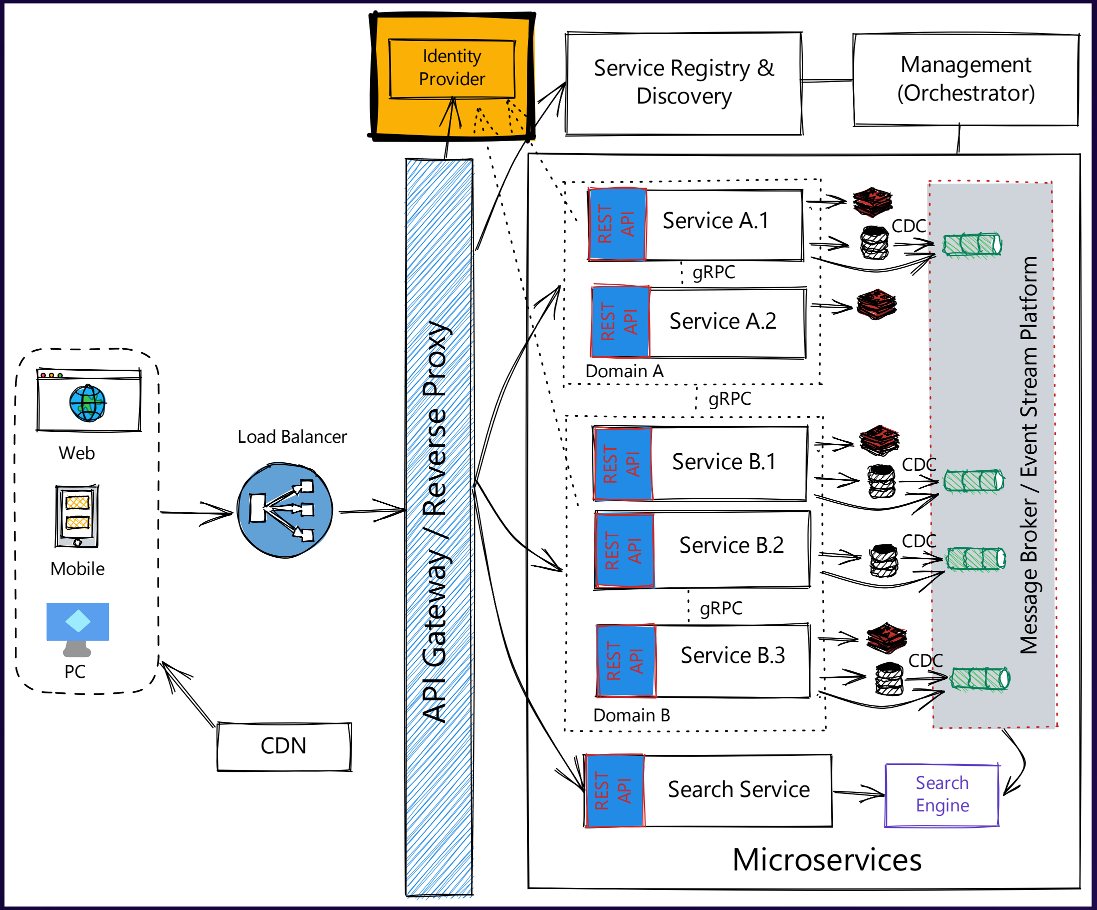


What's new: Performance

- HTTP/2 re-architecture
  - Significant concurrency improvements
  - Server streaming example:
    - .NET 6 – 0.5m RPS
    - .NET 7 – 4.5m RPS (8x increase!)
- gRPC upload speed
  - 108 MB upload with latency:
    - .NET 6 – 26.9 seconds
    - .NET 7 – 4.3 seconds (6x faster!)

https://github.com/LesnyRumcajs/grpc_bench

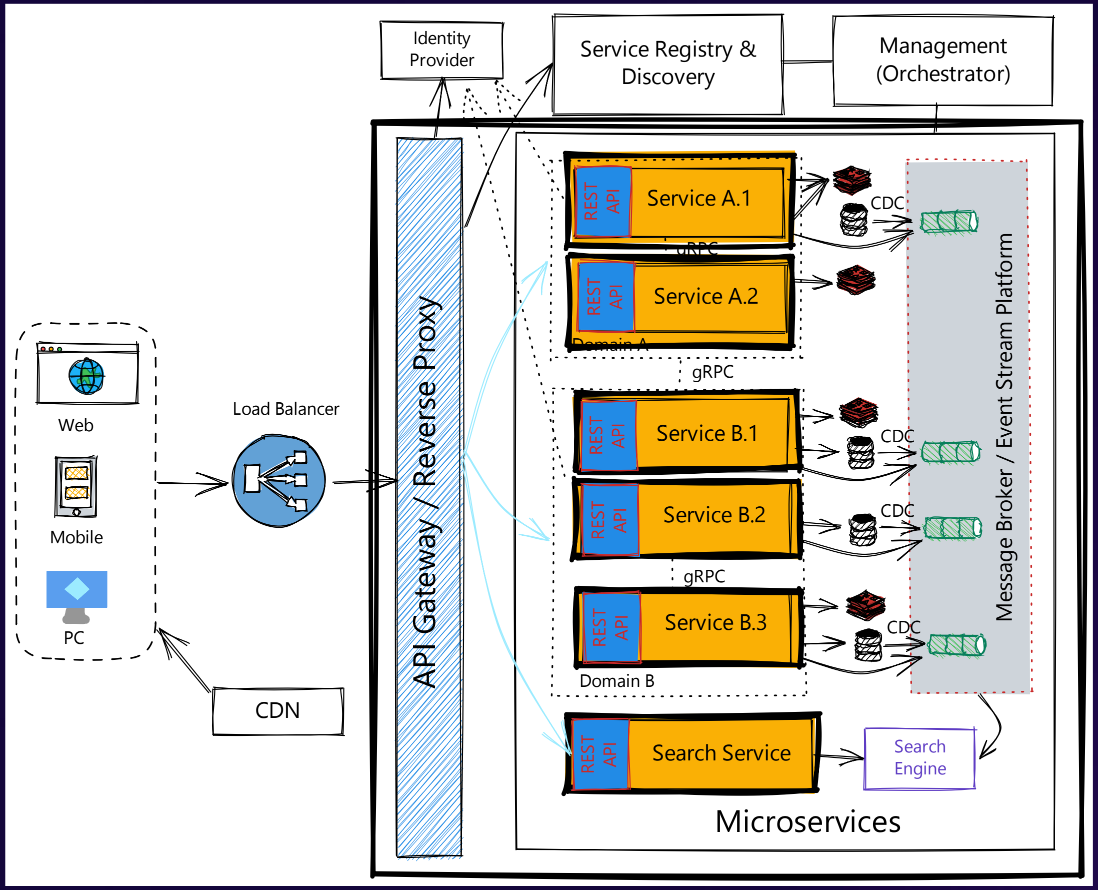
3. Authentication and Authorization



- dotnet user-jwts create => create dev/test JWT
- builder.Services.AddAuthentication().AddJwtBearer() => don't need to register Default schema

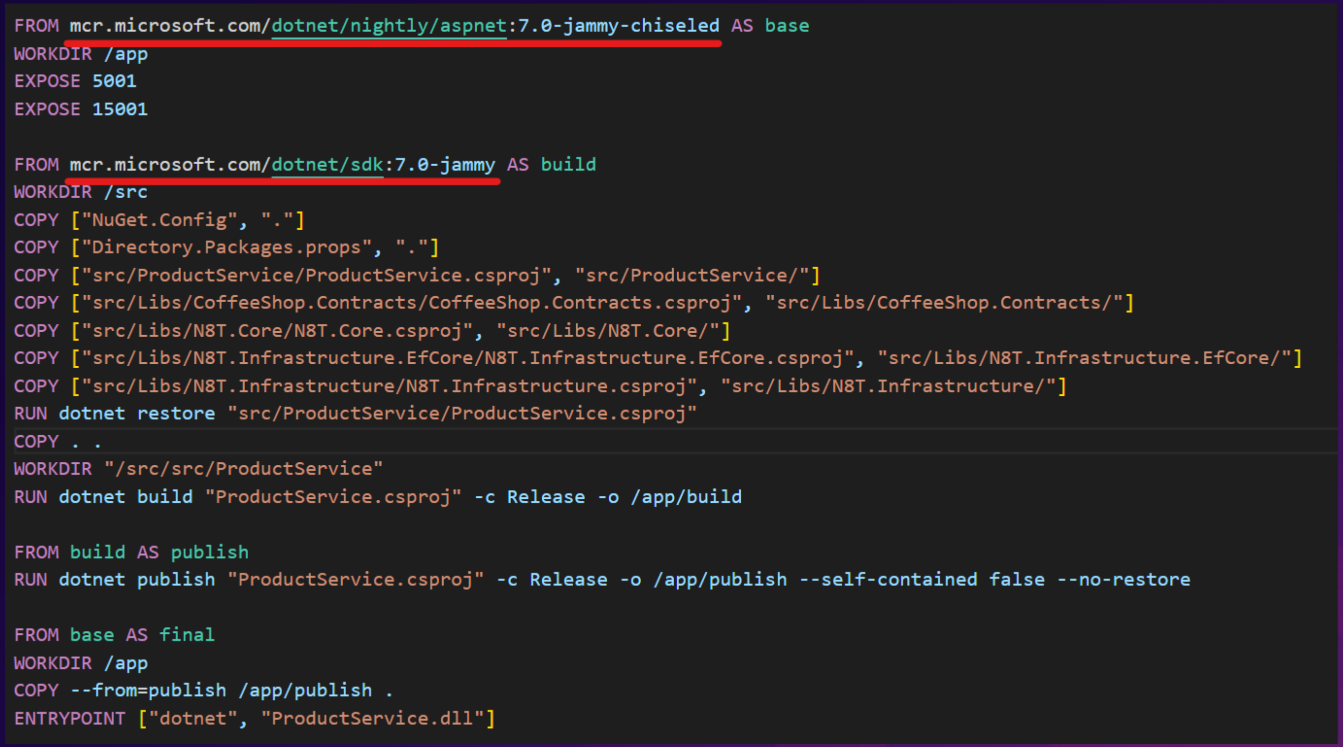
DEMO

4. Docker image size (1/2)

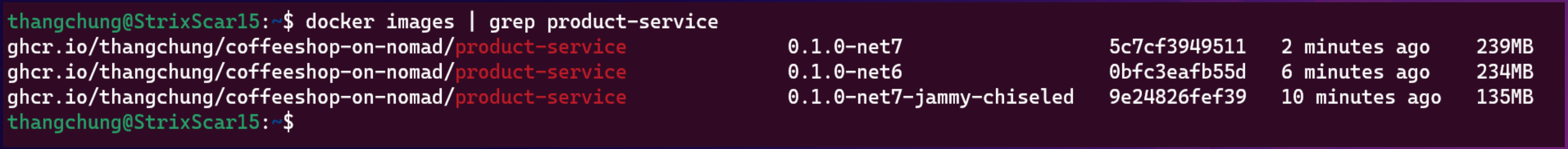


4. Docker image size (2/2)

- .NET 6
- .NET 7
- .NET 7 - Chiseled Ubuntu Containers

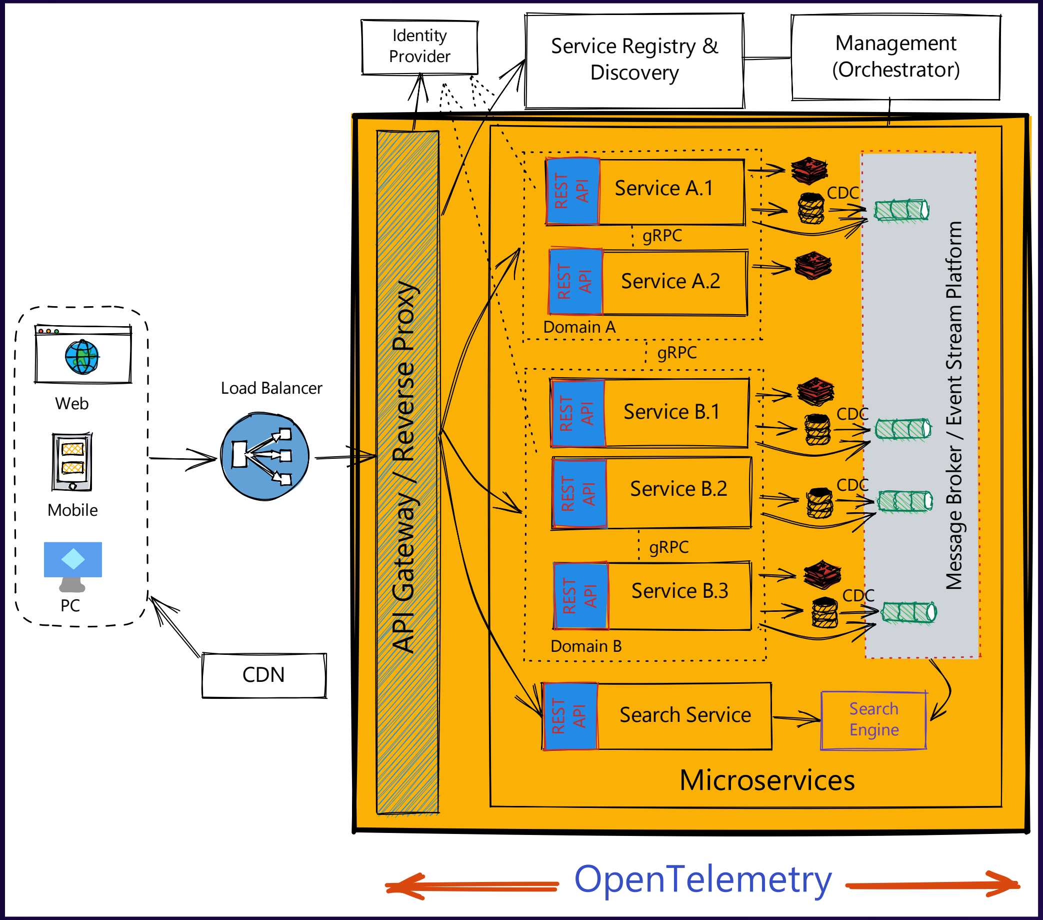


```
FROM mcr.microsoft.com/dotnet/nightly/aspnet:7.0-jammy-chiseled AS base
WORKDIR /app
EXPOSE 5001
EXPOSE 15001

FROM mcr.microsoft.com/dotnet/sdk:7.0-jammy AS build
WORKDIR /src
COPY ["NuGet.Config", "."]
COPY ["Directory.Packages.props", "."]
COPY ["src/ProductService/ProductService.csproj", "src/ProductService/"]
COPY ["src/Libs/CoffeeShop.Contracts/CoffeeShop.Contracts.csproj", "src/Libs/CoffeeShop.Contracts/"]
COPY ["src/Libs/N8T.Core/N8T.Core.csproj", "src/Libs/N8T.Core/"]
COPY ["src/Libs/N8T.Infrastructure.EfCore/N8T.Infrastructure.EfCore.csproj", "src/Libs/N8T.Infrastructure.EfCore/"]
COPY ["src/Libs/N8T.Infrastructure/N8T.Infrastructure.csproj", "src/Libs/N8T.Infrastructure/"]
RUN dotnet restore "src/ProductService/ProductService.csproj"
COPY . .
WORKDIR "/src/src/ProductService"
RUN dotnet build "ProductService.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "ProductService.csproj" -c Release -o /app/publish --self-contained false --no-restore

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "ProductService.dll"]
```



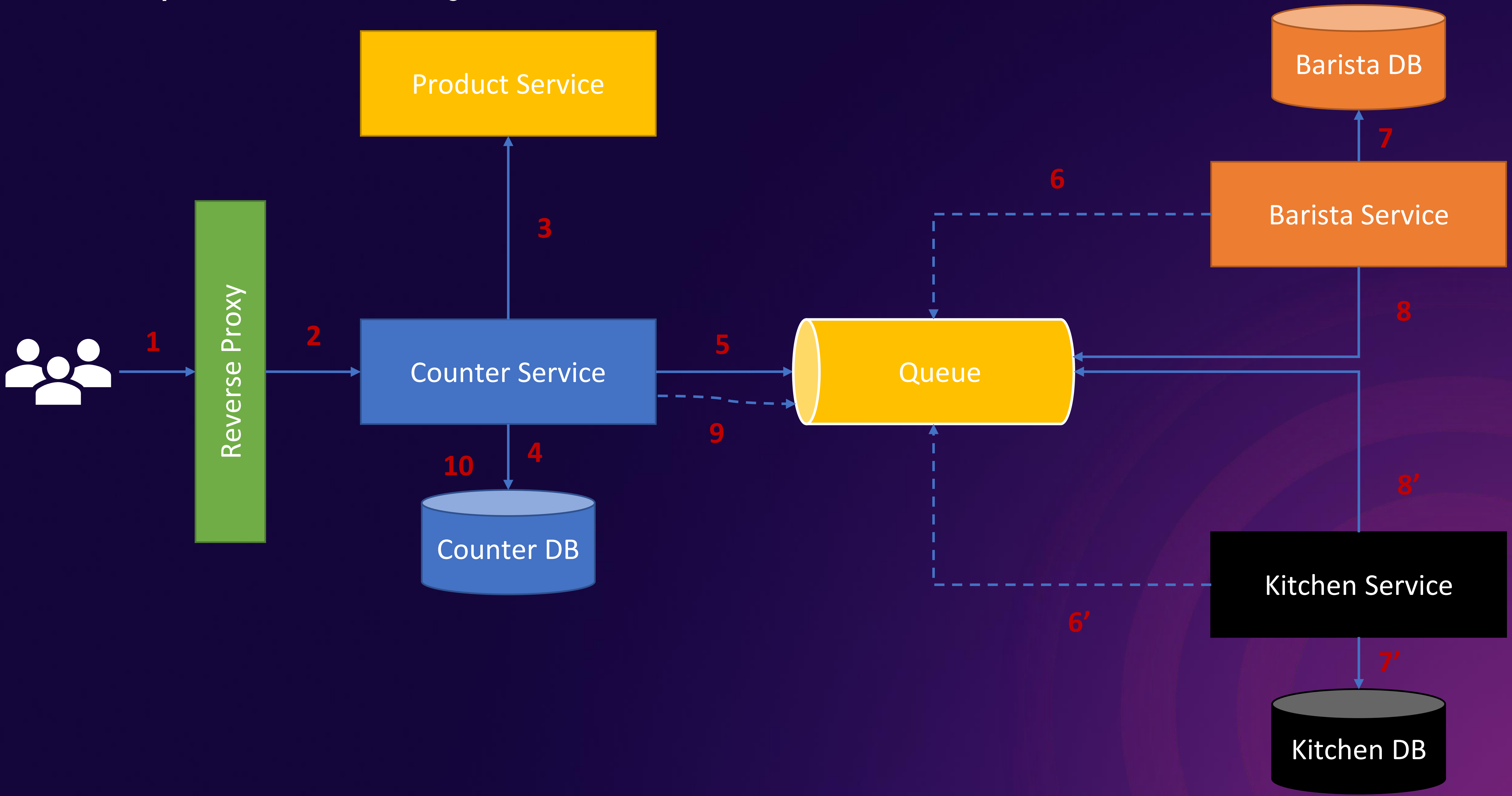
```
thangchung@StrixScar15:~$ docker images | grep product-service
ghcr.io/thangchung/coffeeshop-on-nomad/product-service      0.1.0-net7                  5c7cf3949511   2 minutes ago    239MB
ghcr.io/thangchung/coffeeshop-on-nomad/product-service      0.1.0-net6                  0bfc3eafb55d   6 minutes ago    234MB
ghcr.io/thangchung/coffeeshop-on-nomad/product-service      0.1.0-net7-jammy-chiseled   9e24826fef39   10 minutes ago   135MB
thangchung@StrixScar15:~$
```

5. OpenTelemetry (1/3)

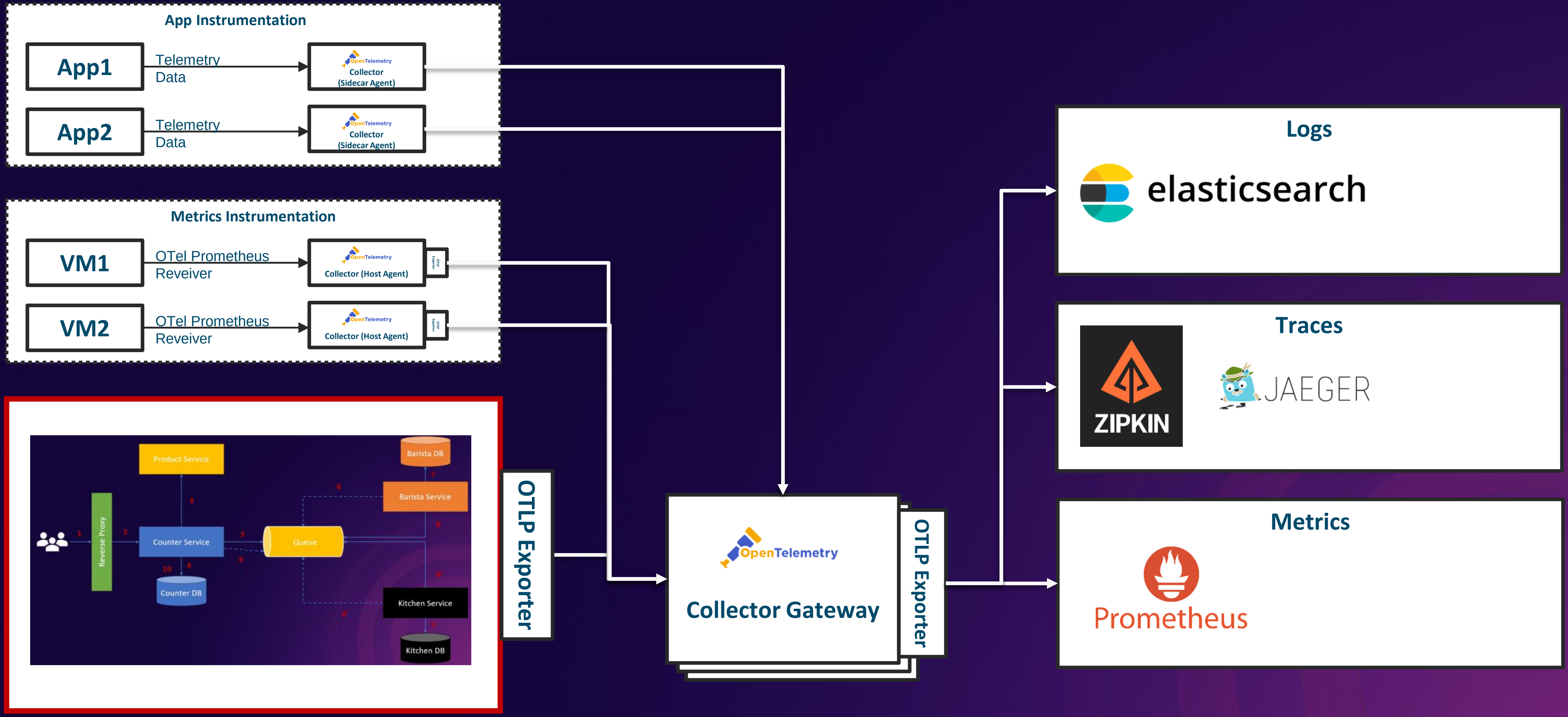


- Logs
- Traces
- Metrics

5. OpenTelemetry (2/3)



5. OpenTelemetry (3/3)



App Instrumentation

App1 → Telemetry Data → OpenTelemetry Collector (Sidecar Agent)

App2 → Telemetry Data → OpenTelemetry Collector (Sidecar Agent)

Metrics Instrumentation

VM1 → OTel Prometheus Reveiver → OpenTelemetry Collector (Host Agent) → OTLP Exporter

VM2 → OTel Prometheus Reveiver → OpenTelemetry Collector (Host Agent) → OTLP Exporter

OTLP Exporter

OpenTelemetry Collector Gateway

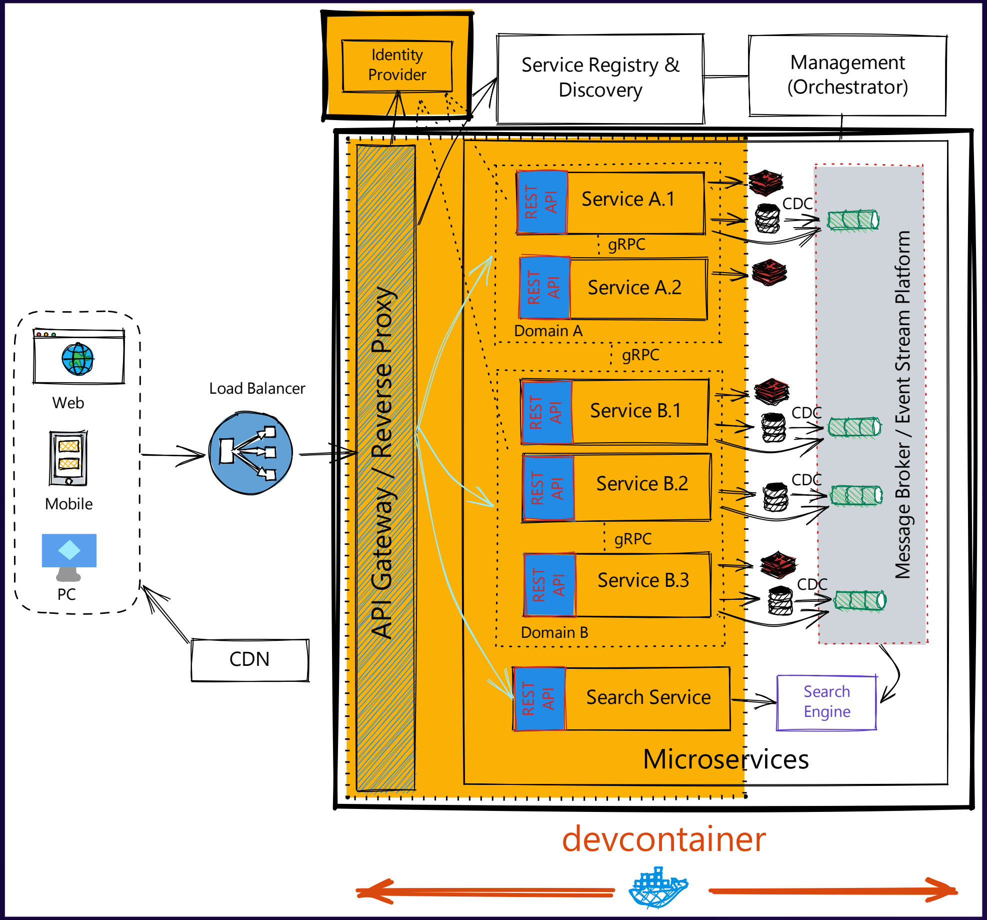
OTLP Exporter

Logs: elasticsearch

Traces: ZIPKIN, JAEGER

Metrics: Prometheus

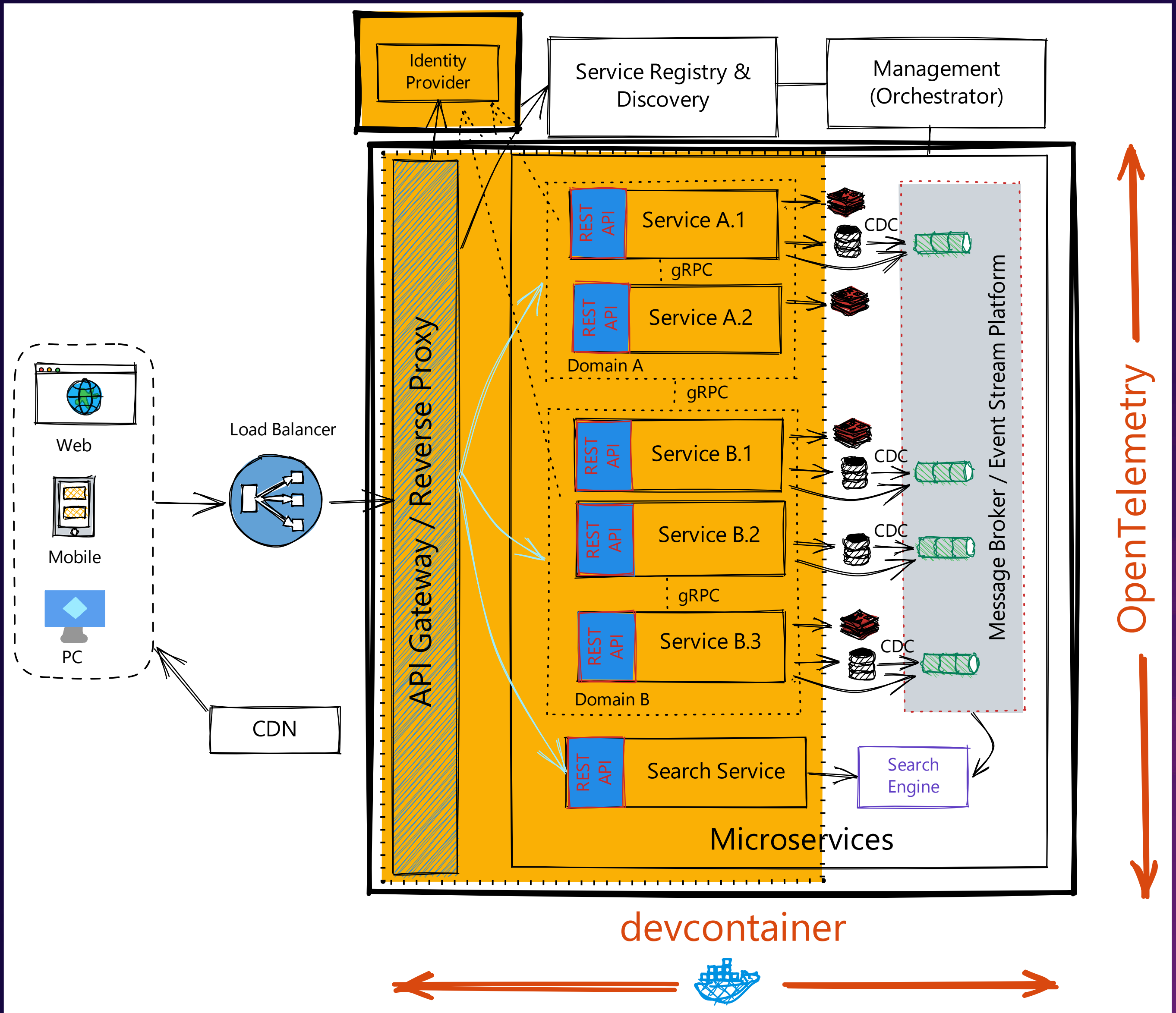
6. Reproducible Development with Devcontainers



- Dev/Test on Docker for Desktop on Windows
- Dev/Test on WSL2 / Ubuntu
- Dev/Test on Github Codespaces

Key takeaways:

- Reverse Proxy (YARP)
- Sync/Async Communication
- AuthN and AuthZ
- Docker image size
- OpenTelemetry
- .devcontainer



Source code: https://github.com/thangchung/coffeeshop-on-nomad/tree/feat/net_conf_2022

Presentation: https://github.com/devcafevn/dotnetconf2022/tree/main/presentations

.NET Conf 2022:

- https://www.youtube.com/@DEVCafeVn
- Microsoft .NET Conf 2022 Day 1: https://www.youtube.com/watch?v=sMOGPRbsbcs&t=1780s
- Microsoft .NET Conf 2022 Day 2: https://www.youtube.com/watch?v=IkL8rJDuB4U

Nash Tech.

Thanks for joining!

DEV CAFE
Connect and share

https://www.facebook.com/devcafevn
https://www.youtube.com/c/devcafevn