Implementation

Implementation and container environment guide (Implementation Specifics)

The comprehensive technical document for architectural design and code structure

Engineering & Operations Team (WSS)

Technical Application Profile

Project : NotificationPublisher Ecosystem

Paths (Endpoints) : Email (Single/Bulk), SMS, WhatsApp, FCM

Data protection (Secrets) : HashiCorp Vault KV v2

Implementation stage : Production - OKE Cluster

Core Tech : ASP.NET Core 8.0, RabbitMQ

1. Local Docker Compose (Local Provisioning)

To ease the development cycle, docker-compose.local.yml integrates pods for 6 services.

- Network provisioning: Docker creates notificationgateway_default as an isolated DNS space
- Starting RabbitMQ: The broker starts on port 5672 with Health Check
- Vault injection: vault-agent-local loads sensitive variables and writes them to a shared Docker volume
- Dependencies: .NET containers do not start until Vault completes successfully (Condition: service_completed_successfully)

```
  notification-api:
    build:
      context: ./Notification-Api
      dockerfile: Dockerfile
    container_name: notification-api-local
    ports:
      - "8082:8080"
    volumes:
      - ./Notification-Api/config:/app/config:ro
    depends_on:
      vault-agent:
        condition: service_completed_successfully
```

2. ASP.NET Core Configuration (Startup.cs Configuration)

Register API version

```
services.AddApiVersioning(config => {
    config.DefaultApiVersion = new ApiVersion(Major, Minor);
    config.AssumeDefaultVersionWhenUnspecified = true;
    config.ReportApiVersions = true;
})
.AddApiExplorer(options => {
    options.GroupNameFormat = "'v'VVV";  // v1, v1.1, v2
    options.SubstituteApiVersionInUrl = true;
});
```

Service registration strategy

NotificationPublisher - Implementation

```
// Dynamic attribute-based registration
services.AddServicesWithAttributeOfType<ScopedServiceAttribute>();

// Explicit registration
services.AddScoped<IS3StorageService, S3StorageService>();
services.AddHttpClient();  // Pooled HttpClient factory

// Authentication scheme
services.AddAuthentication("BasicAuthentication")
    .AddScheme<AuthenticationSchemeOptions,
        BasicAuthenticationHandler>("BasicAuthentication", null);

// Ephemeral data protection (stateless containers)
services.AddDataProtection()
    .UseEphemeralDataProtectionProvider();
```

- Dynamic registration: Any class decorated with [ScopedService] is registered automatically
- HttpClient Factory: Prevents socket exhaustion under high concurrency
- S3 Storage: Scoped to ensure AmazonS3Client is initialized for each request
- Temporary data protection: Keys do not persist across container restarts

3. HTTP Pipeline Order

The request pipeline is configured with a precise order that defines how each request is processed:

```
// Startup.cs Configure()
app.UseMiddleware<SwaggerBasicAuthMiddleware>();  // 1
app.UseSwagger();                                 // 2
app.UseSwaggerUI();                               // 3
app.UseRouting();                                 // 4
app.UseCors("NotificationGatewayUI");             // 5
app.UseAuthentication();                          // 6
app.UseAuthorization();                           // 7
app.UseEndpoints(e => e.MapControllers());        // 8
```

Job	Step
Swagger intercepts and enforces Basic Auth against unauthorized access	1. SwaggerAuth
Provides the OpenAPI JSON specification	2. UseSwagger
It offers a Swagger UI HTML interface	3. SwaggerUI
Matches incoming URLs to controller paths	4. UseRouting
Enforces the CORS open interface policy	5. UseCors
BasicAuthenticationHandler ينفذ	6. Authentication
Enforces [Authorize] attributes on controllers	7. Authorization
It sends the request to the corresponding controller	8. MapControllers

4. Configure S3 Storage Service

```
var config = new AmazonS3Config {
    ServiceURL = PublisherSetting.S3ServiceURL,
    ForcePathStyle = true  // Non-AWS compatibility
};

_s3Client = new AmazonS3Client(
    PublisherSetting.S3AccessKey,
    PublisherSetting.S3SecretKey,
    config
);
```

Automatic content-type detection

MIME type	Extension
application/pdf	pdf
application/msword	doc
application/vnd.openxmlformats-...wordprocessing	docx
application/vnd.ms-excel	xls
application/vnd.openxmlformats-...spreadsheet	xlsx
image/png	png
image/jpeg	jpg / jpeg
image/gif	gif
video/mp4	mp4
video/x-msvideo	avi
video/quicktime	mov
application/octet-stream	default

5. Dependency Injection Lifecycle (DI Lifecycle)

Examples	Behavior	Style
RabbitMQ Connection	One copy for the life of the application	Singleton
PublisherOperations, DapperHelper	A new version for each HTTP Request	Scoped
SMSConsumer.Worker	Infinite loop via IHostedService	Background