# Docker-Based Excel Template Management & Configuration Externalization (Template as Application Asset)

---

## 1. Purpose

This document defines a **clean Docker image–based approach** for managing Excel templates used by .NET microservices, while **retaining Azure Blob Storage exclusively for generated reports**.

It also formalizes a **required configuration improvement** to externalize the Azure Blob `Report` **container name** from hardcoded constants into application configuration.

## 2. Current State (Baseline)

### Existing Behavior

- Excel templates are stored in **Azure Blob Storage → `template` container**
- Excel reports are stored in **Azure Blob Storage → `report` container**
- Template container name is implicitly hardcoded in services
- Report container name is read from `AppConstants.cs`
- .NET microservices:
    - Fetch templates from Blob
    - Populate using Aspose / .NET libraries
    - Upload generated reports back to Blob
- Angular UI downloads reports via APIs

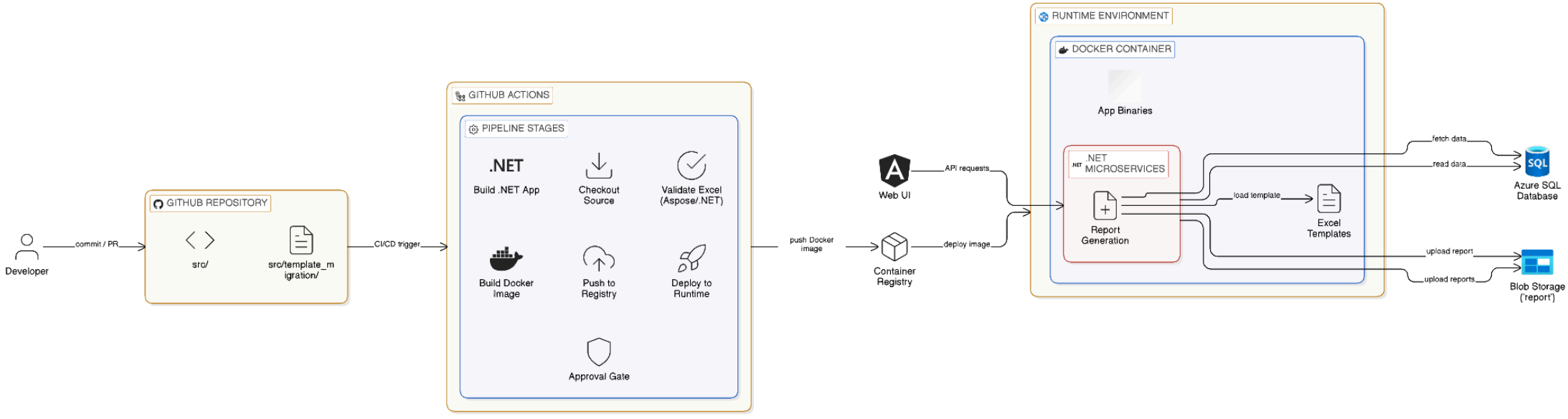
## 3. Goal

### Key Architectural Decisions

1. **Excel templates will be packaged inside Docker images**
2. **Azure Blob Storage will no longer be used for templates**
3. **Azure Blob Storage will continue to be used for reports**
4. **All container names must be read from configuration (AppSettings)**

Templates become **application assets**, not external configuration.

# 4. Updated Architecture Overview



# 5. Repository Structure

```
repo-root/
│
├── src/
│   ├── PipelineService/ReportService
│   │   ├── Program.cs
│   │   ├── AppConstants.cs   (to be reduced)
│   │   └── Dockerfile
│   │
│   └── template_migration/
│       ├── TaxReport.xlsx
│       ├── PayrollSummary.xlsx
│       └── ComplianceAudit.xlsx
│
```

# 6. Developer Responsibilities

## 6.1 Template Management

Developers modify Excel templates **only** under:
src/template_migration/

- 
- No templates are stored or edited in Azure Blob Storage
- Any template change requires:
    - Git commit
    - PR review
    - Docker rebuild

## 6.2 Required Code Change – Template Access

**Current (Blob-Based) - it will be removed**

GetBlobClient("template/TaxReport.xlsx");

**Target (Docker File System) - it will be added**

/app/templates/TaxReport.xlsx

## 6.3 Configuration-Driven Template Access

**appsettings.json**

```
{
  "TemplateSettings": {
    "Provider": "FileSystem",
    "BasePath": "/app/templates"
  }
```

}

**Code (One-Time Change)**

```
var basePath = configuration["TemplateSettings:BasePath"];
var templatePath = Path.Combine(basePath, $"{templateName}.xlsx");

using var stream = File.OpenRead(templatePath);
```

This replaces all template Blob reads.

# 7. Required Code Change – Report Container Name

## 7.1 Problem Statement

- Currently, the Azure Blob **report container name** is:
  - Hardcoded in `AppConstants.cs`
  - Repeated across multiple services
- This violates configuration best practices and complicates environment-specific deployments.

## 7.2 Target State (Correct)

All Azure Blob container names “report”  must be read from configuration.

## 7.3 Configuration Change

**`appsettings.json`**

```
{
  "StorageSettings": {
    "ReportContainerName": "report"
  }
}
```

## 7.4 Code Change

**Current - to be removed**

```
BlobContainerClient client =
    blobServiceClient.GetBlobContainerClient(AppConstants.ReportContainer);
```

**Target - to be added**

```
var containerName = configuration["StorageSettings:ReportContainerName"];
BlobContainerClient client =
    blobServiceClient.GetBlobContainerClient(containerName);
```

### 7.5 Impact

| Area | Impact |
|---|---|
| Code change | Minimal |
| Risk | Low |
| Benefit | High |
| Env flexibility | Enabled |

This change should be applied **across all microservices**.

# 8. Dockerfile - to be done by devops team

```
FROM mcr.microsoft.com/dotnet/aspnet:8.0

WORKDIR /app

COPY ./publish/ .
COPY ./src/template_migration /app/templates
```

# 9. DevOps Responsibilities

### DevOps Will:

1. Update Dockerfile to embed templates
2. Update CI/CD pipeline:
    - Rebuild image when `src/template_migration` changes
3. Remove Azure Blob:
    - `template` container
    - Template-related access permissions
4. Retain Azure Blob for:
    - `report` container only
5. Ensure environment-specific config:
    - `ReportContainerName`
    - Storage connection strings

# 10. CI/CD Behavior

| Change | Action |
|---|---|
| Template change | Docker rebuild + redeploy |

| | |
|---|---|
| Code change | Docker rebuild + redeploy |
| Config change | Restart / redeploy |
| Report logic | No template impact |

# 11. Runtime Behavior

| Artifact | Source |
|---|---|
| Excel templates | Docker image filesystem |
| Generated reports | Azure Blob Storage (`report`) |
| Data | Azure SQL |
| UI downloads | API + Blob |

---

# 12. Rollback Strategy

- Rollback = redeploy previous Docker image
- Templates automatically roll back
- No Blob cleanup required

# 13. Security & Governance

| Area | Control |
|---|---|
| Templates | Immutable inside image |
| Report storage | Blob RBAC |
| CI/CD | Approval gates |
| Audit | Git + image tags |

# 14. Trade-Off Summary

### Advantages

- Removes template Blob dependency
- Strong runtime immutability
- Simplified infrastructure
- Predictable deployments

### Disadvantages

- Template change requires redeploy
- Larger Docker images
- Slower rollback than Blob-only

# 16. Final Recommendation

This approach is **technically sound and clean** if the organization is comfortable with:

- Redeploying services for template changes
- Treating templates as part of application binaries

The configuration change for **report container name** is **recommended regardless of which template strategy is chosen**.