Deployment

Advanced Distribution Architecture (Kubernetes & CI/CD)

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. Launch technologies via Azure Pipelines

The deployment strategy relies on the Notification-pipelines.yml file to build containers and deploy to OKE.

- Azure DevOps running runner on Linux Agent Pool (Linux-Agent-KeycloakVM)
- Parameters inject ASPNETCORE_ENVIRONMENT (Staging or Testing)
- Runner loads oke-kubeconfig certificates to access the cluster

Multiple environment settings

Testing	Staging	Parameter
Testing	Staging	ASPNETCORE_ENVIRONMENT
api/dev	api/staging	Vault KV Path
testing	staging	Kubernetes Namespace
testing-{buildId}	staging-{buildId}	OCIR Image Tag

2. Multi-Stage Docker Build

```
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
WORKDIR /src
COPY ["EmailConsumer/EmailConsumer.csproj", "EmailConsumer/"]
RUN dotnet restore "EmailConsumer/EmailConsumer.csproj"
COPY .
RUN dotnet publish -c Release -o /app/publish

FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS runtime
WORKDIR /app
COPY --from=build /app/publish .
ENTRYPOINT ["dotnet", "EmailConsumer.dll"]
```

Compare image size

Purpose	Size	Photo
Compilation, NuGet restore	~800MB	dotnet/sdk:8.0
Runtime-only, stripped	~180MB	dotnet/aspnet:8.0

3. StatefulSet ⅃ RabbitMQ

- RabbitMQ is deployed as a StatefulSet (not a Deployment) to ensure stability
- Stable Identity: Each Pod gets a stable name (rabbitmq-0)

- Ordered operations: Pods are created and deleted sequentially to prevent split-brain
- Persistent NFS storage: Queue data is mounted on an NFS volume that can withstand reboots

```
# Deployments: standard rollout
kubectl rollout restart deployment/notification-api

# StatefulSets: requires explicit handling
kubectl rollout restart statefulset/rabbitmq
```

4. Post-Deployment Verification

Success criterion	method	Examination
All Pods are in a Running state	kubectl get pods	Pod Readiness
Management UI responds on 15672	Health check endpoint	RabbitMQ
Returns 200 (after authentication)	HTTP GET /swagger	API Health
No crash loop errors	kubectl logs	Consumer Status