CONTAINER TOPOLOGY DIAGRAM

Core Digital Transformation for Retail Banking

Lab: Container, Docker, and IBM Cloud Container Registry · Task 1

Date: Feb 18, 2026	Version: 1.0	Prepared By: Solution Architect	Status: Draft

Overview

This document presents the initial container topology for three core banking microservices deployed on a cloud-native, containerized infrastructure. The architecture segments services into a public-facing network (API Gateway) and an internal network (Auth, Loan, and Onboarding services), with container images stored in and pulled from a cloud container registry (IBM Cloud Container Registry or AWS ECR).

Microservices Summary

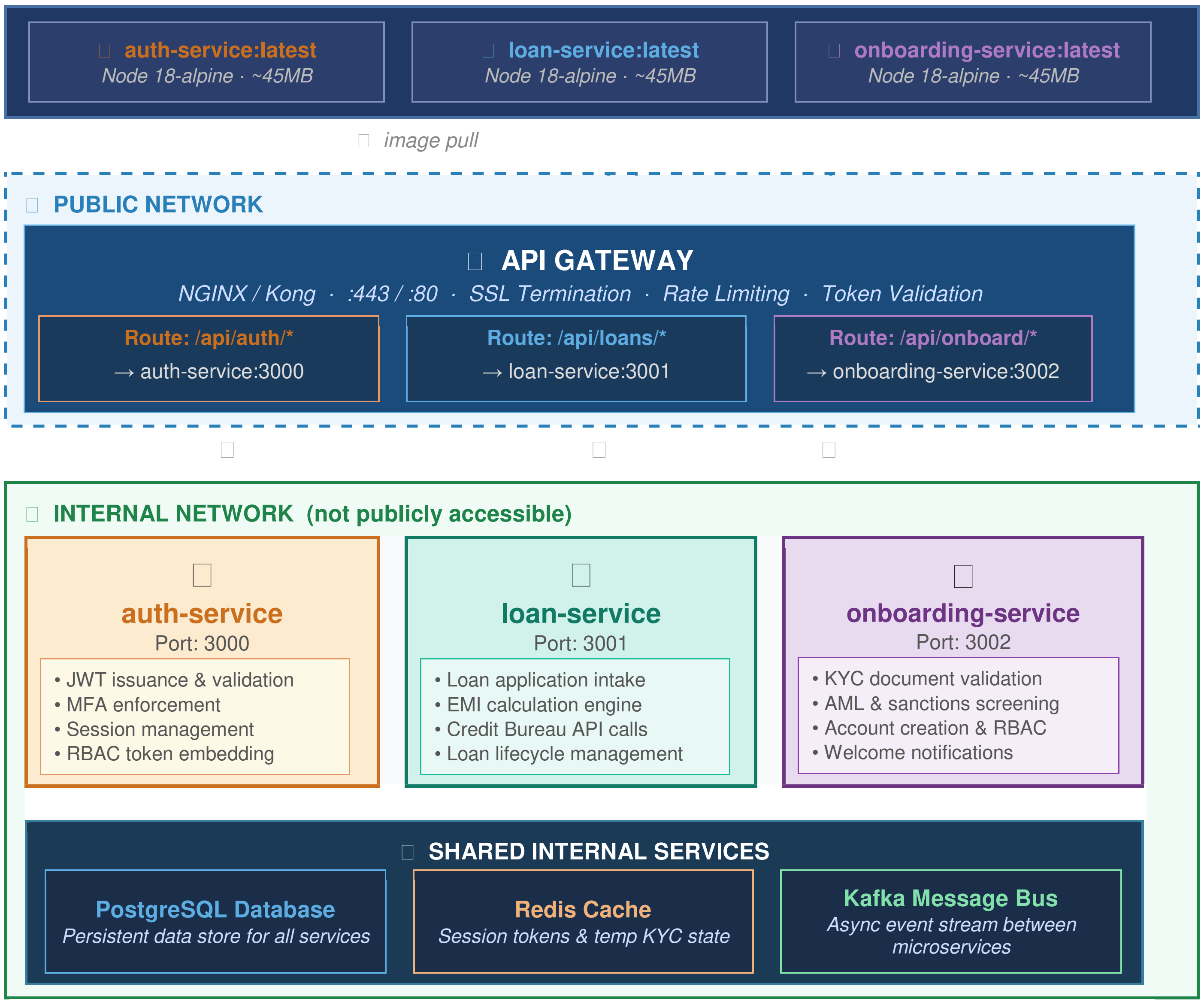
Service	Responsibility	Network Zone	Exposed Port
API Gateway	Routes external requests to internal microservices; SSL termination, rate limiting, auth token validation	□ **Public Network**	443 / 80
auth-service	Handles customer login, JWT token issuance, MFA verification, and session lifecycle management	□ **Internal Network**	3000
loan-service	Processes loan applications, EMI calculations, credit bureau API calls, and manages the loan lifecycle	□ **Internal Network**	3001
onboarding-service	Validates KYC documents, opens new customer accounts, triggers AML screening, and sends welcome notifications	□ **Internal Network**	3002

Container Topology Diagram

The diagram below represents the deployment-level architecture. Dashed blue borders indicate the Public Network zone; solid green borders indicate the Internal Network zone. Arrows show request flow and image-pull direction from the container registry.

☁ CONTAINER REGISTRY

IBM Cloud Container Registry · AWS ECR (Elastic Container Registry)



Network Segmentation

Zone	Services	Purpose & Exposure
Public Network	API Gateway (NGINX / Kong) · Port 443, 80	Exposed to internet. Handles SSL termination, auth token validation, rate limiting, and routes to internal services via reverse proxy.
Internal Network	**auth-service · Port 3000** **loan-service · Port 3001** **onboarding-service · Port 3002**	Not publicly accessible. Services communicate only via the API Gateway or internal Kafka event bus. All inter-service calls stay within the private VPC/network namespace.

Security Best Practices & Compliance

Practice	Implementation Detail
Image Signing	All container images are signed using Docker Content Trust (DCT) before being pushed to IBM Cloud Container Registry / AWS ECR, ensuring image integrity and preventing tampered images from running.
Role-Based Access (RBAC)	Each microservice runs as a non-root user with least-privilege permissions. Kubernetes RBAC policies restrict which pods can access secrets, config maps, and inter-service APIs.
Vulnerability Scanning	IBM Cloud Container Registry and AWS ECR are configured to automatically scan all pushed images for CVEs using built-in vulnerability advisors (Trivy / Clair). Builds with critical CVEs are blocked.
Network Policies	Kubernetes NetworkPolicy objects enforce that only the API Gateway can reach internal services. auth-service, loan-service, and onboarding-service cannot be reached directly from outside the cluster.
Secrets Management	No secrets or API keys are embedded in Dockerfiles or environment variables. All secrets are injected at runtime via Kubernetes Secrets or IBM Secrets Manager / AWS Secrets Manager.
Compliance Standards	Architecture is aligned with PCI-DSS (cardholder data isolation), GDPR (data residency in approved regions), AML (audit logging on all transactions), and KYC (identity verification step in onboarding-service).

Validation Checklist

Validation Item	Status	Notes
Three microservices defined: auth-service, loan-service, onboarding-service	✓ Yes	*All containers specified with ports and responsibilities*
API Gateway included and connected to all three services	✓ Yes	*Routes /auth, /loans, /onboard namespaces*
Public network zone clearly separated from internal network	✓ Yes	*Dashed blue = public, solid green = internal*
Container registry shown with image-pull arrows to services	✓ Yes	*IBM Cloud CR / AWS ECR with three image repos*
Security best practices documented (signing, RBAC, scanning)	✓ Yes	*Six security controls defined*
Shared internal services layer included (DB, Cache, Messaging)	✓ Yes	*PostgreSQL, Redis, Kafka*

Regulatory compliance alignment noted (PCI-DSS, GDPR, AML, KYC)	✓ **Yes**	*Mapped to onboarding and data isolation controls*
Diagram saved / exportable as container_topology_diagram.pdf	✓ **Yes**	*Use Lucidchart / Draw.io to render full visual*