# Project Blueprint Hotel Application – Java Spring Boot + AWS (MySQL + MongoDB) Project Blueprint

**1. Vision & Goals**

* **Multi-property hotel platform** at enterprise scale.
* **Core flows**: search → availability → dynamic pricing → book → pay → check-in/out.
* **Operational modules**: housekeeping, maintenance, rate management, loyalty, reporting.
* **Target cloud**: AWS (EKS, RDS, DocumentDB, S3, MSK/SQS/SNS).
* **Database strategy**:
  + MySQL → transactional, ACID (reservations, catalog, payments).
  + MongoDB → unstructured/event data (audits, tasks, notifications).
* **Scalability goals**: Cloud-native, containerized, CI/CD, observability, zero-downtime deployments.

**2. Personas & User Stories**

* **Guest**: search, book, pay, manage reservations.
* **Front Desk Agent**: check-in/out, assign rooms, upsell.
* **Housekeeping**: manage room status, tasks.
* **Revenue Manager**: configure rates, promotions.
* **Admin/IT**: manage properties, roles, tenants, integrations.

**3. High-Level System Architecture**

**Frontend**

* Angular SPA (guest + back-office).

**Backend (Spring Boot Microservices)**

1. **Identity & Access** – Authentication, authorization, JWT.
2. **Catalog Service** – Property, room type, amenities.
3. **Inventory Service** – Room lifecycle, housekeeping linkage.
4. **Rates & Restrictions Service** – Rate plans, calendars, promotions.
5. **Dynamic Pricing Service** – Rate suggestions (ML-ready).
6. **Availability Service** – Combine inventory + rates + pricing + restrictions.
7. **Booking Service** – Reservation lifecycle.
8. **Payments Service** – Payment orchestration with providers.
9. **Housekeeping Service** – Tasks and room status updates.
10. **Notifications Service** – Email/SMS/push.
11. **Search Service** – Property/room search + recommendations.
12. **Reporting Service** – Pre-aggregated KPIs and reports.

**Shared Infrastructure**

* API Gateway → Spring Cloud Gateway.
* Event Bus → Kafka (MSK) or RabbitMQ (AWS MQ).
* Config/Secrets → Spring Cloud Config + AWS Secrets Manager.
* Observability → Prometheus, Grafana, OpenTelemetry, CloudWatch.

**4. Data Strategy**

**MySQL (RDS/Aurora)**

* Property, RoomType, Room
* RatePlan, Rate, Restriction
* Reservation, ReservationLine
* Payment, Invoice, Guest

**MongoDB (DocumentDB)**

* HousekeepingTask
* NotificationsTemplate
* BookingAudit
* Promotions
* Search index documents

**Cache:** Redis (ElastiCache) for availability snapshots.  
**Storage:** S3 for media, invoices, exports.

**5. Event Model (Domain Events)**

* **ReservationCreated** → triggers confirmation email, housekeeping prep.
* **PaymentSucceeded** → updates booking, generates invoice.
* **RoomStatusChanged** → updates housekeeping dashboard.
* **RateSuggested** → from Pricing to Rates service.

Events are immutable JSON/Avro published via Kafka/SNS.

**6. AWS Cloud Deployment**

* **Compute:** Amazon EKS (Kubernetes)
* **Databases:**
  + Aurora MySQL → transactional data
  + DocumentDB → NoSQL documents
* **Messaging:**
  + MSK (Kafka) for event streaming
  + SQS/SNS for lightweight messaging
* **Cache:** ElastiCache Redis
* **Storage:** S3 (images, invoices, exports)
* **Identity:** Amazon Cognito (JWT)
* **Secrets:** AWS Secrets Manager
* **Observability:** CloudWatch + OpenTelemetry + Grafana

**7. Security & Compliance**

* **Authentication:** OAuth2/OIDC with JWT.
* **Authorization:** RBAC + tenant isolation per property/brand.
* **Data security:**
  + PII minimization
  + TLS in transit, encryption at rest (KMS/TDE)
* **Payments:** PCI DSS reduction via provider tokenization.
* **App security:**
  + Input validation, CSRF protection, rate limiting, audit trails.

**8. DevOps & CI/CD**

* **Source Control:** GitHub mono-repo.
* **CI/CD:** GitHub Actions → Docker → push to AWS ECR → deploy via Argo CD/Flux to EKS.
* **IaC:** Terraform (VPC, EKS, RDS, DocumentDB, ElastiCache, MSK, Secrets Manager).
* **Secrets management:** no plaintext secrets; injected at runtime.
* **Deployment strategy:** blue/green or canary rollouts.

**9. Observability & SRE**

* **Logging:** Structured logs → CloudWatch/Loki.
* **Metrics:** Micrometer → Prometheus → Grafana dashboards.
* **Tracing:** OpenTelemetry → AWS X-Ray / Grafana Tempo.
* **SLOs:**
  + Booking success rate > 99%
  + Search latency p95 < 300ms

**10. Test Strategy**

* **Unit:** JUnit + Mockito.
* **Integration:** Testcontainers (MySQL, Mongo, Kafka).
* **Contract:** Pact (consumer-driven contracts).
* **E2E:** Playwright (search → book → pay → confirm).
* **Performance:** k6 load tests.
* **Chaos:** Pod kill, latency injection in staging.

## 11. Iterative Delivery Plan (8-Week Sprint Roadmap)

* **Sprint 0 (Prep)** → Infra, repo skeleton, CI/CD pipelines, local dev setup.
* **Sprint 1** → Identity Service + Property Catalog Service (CRUD, MySQL, basic APIs).
* **Sprint 2** → Inventory Service + Rates & Restrictions Service (room lifecycle, seasonal rates).
* **Sprint 3** → Availability Service (combine inventory + rates + pricing, Redis caching).
* **Sprint 4** → Booking Service (reservation lifecycle) + Payments Service (Stripe sandbox integration).
* **Sprint 5** → Notifications Service (email confirmations, SendGrid/Twilio) + basic Angular Guest Booking Flow.
* **Sprint 6** → Housekeeping Service (tasks, room status updates) + Front Desk flows (check-in/out).
* **Sprint 7** → Dynamic Pricing Service (rule-based seasonal multipliers) + Search Service (basic property/room search with MongoDB).
* **Sprint 8** → Reporting Service (pre-aggregated KPIs in MySQL) + Observability (Prometheus, Grafana, OpenTelemetry, CloudWatch dashboards) + Blue/Green Deployment readiness.  
    
    
  12. Team Setup & Developer Environment
* **IDE:** IntelliJ IDEA (Ultimate recommended for DB & K8s integration)
* **Local tools:**
  + JDK 21, Maven 3.9+
  + Docker Desktop, kubectl, Helm, AWS CLI
  + Postman, DBeaver, MailHog
  + Optional: K9s (Kubernetes UI), ngrok (tunnels), Terraform CLI
* **Local stack via docker-compose:**
  + MySQL, MongoDB, Redis, Kafka, MailHog

**13. Cost-Optimized Setup (Dev/Non-Prod)**

* Local-first with Docker Compose.
* AWS free tiers where possible:
  + RDS Aurora serverless v2 (pause when idle)
  + S3 for media
  + Cognito free tier for auth
  + Grafana Cloud free for metrics/logs
* Avoid managed Kafka until necessary; start with RabbitMQ/SQS.

14. Folder & Repo Structure (Mono-Repo)  
  
/hotel-platform

/apps

/web-guest (angular)

/web-backoffice (angular)

/services

/identity (spring-boot, mysql)

/catalog (spring-boot, mysql)

/inventory (spring-boot, mysql)

/rates (spring-boot, mysql)

/availability (spring-boot, mysql + redis)

/booking (spring-boot, mysql + mongo)

/payments (spring-boot, mysql)

/housekeeping (spring-boot, mongo)

/notifications (spring-boot, mongo)

/pricing (spring-boot, mysql + mongo)

/search (spring-boot, mongo)

/reporting (spring-boot, mysql)

/deploy

/terraform (infra modules)

/k8s (helm charts)

/shared

/contracts (avro/json schemas for events)

/libs (shared Java libs)

/docs

architecture.md

api-specs.md

runbooks/