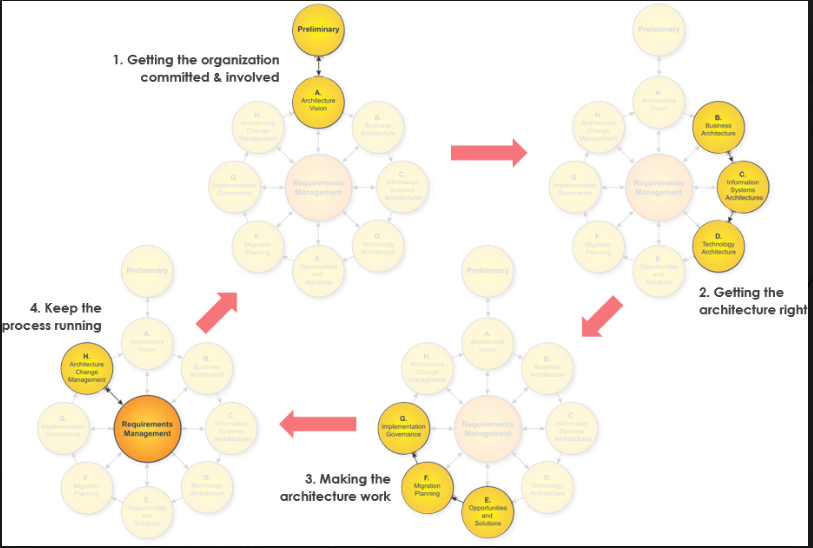
**"In our greenfield retail e-commerce project, I followed the TOGAF ADM framework to define, design, and deliver a scalable, secure, and resilient architecture on Azure using .NET Core microservices. Here's how I applied each phase in the real-world project lifecycle.**

**"*I apply TOGAF for strategic architecture — identifying business capabilities, defining target state, and creating a roadmap. But during project implementation, I apply Azure Well-Architected Framework to align our workloads to cloud-native best practices across reliability, security, and performance.*"**

**"*TOGAF gave me what to build and why, Azure WAF helped me define how to build it right in Azure.*"**

**TOGAF is strategic, WAF is tactical"**

**✅ TOGAF + Azure/.NET Retail Architecture Roadmap (Full Lifecycle)**

****

|  |  |  |  |
| --- | --- | --- | --- |
| TOGAF Phase | Strategy (What You Did) | Tools Used | NFRs Addressed |
| Preliminary | **Initiated architecture request and identified stakeholders** | **Confluence, Email** | **–** |
| A. Architecture Vision | **Defined high-level model, goals, and NFRs** | **Draw.io, Confluence** | **Captured: Scalability, Security, RTO, RPO** |
| B–D. Design (Business, App/Data, Technology Architecture) | **DDD-based microservices (Product, Cart, Order, Payment), Azure tech stack selection** | **Visual Studio, Draw.io, Lucidchart, Confluence** | **- Scalability: AKS, Cosmos DB - Security: Azure AD B2C, OAuth2, Key Vault - Resilience: Circuit breakers, Retry policies - Observability: Azure Monitor, App Insights** |
| E. Opportunities & Solutions | **Mapped reusable Azure services and proposed cost-optimized building blocks** | **Azure Pricing Calculator, Excel, Confluence** | **Faster time-to-market, lower cost** |
| F. Migration Planning | **Iterative rollout plan with blue-green deployment** | **Azure DevOps Boards, Confluence** | **Zero downtime releases (HA/DR)** |
| G. Implementation Governance | **Setup architecture gates, code quality checks** | **Azure DevOps Pipelines, SonarQube, OWASP ZAP** | **Quality, Compliance, Security** |
| H. Architecture Change Management | **Handled evolving needs (e.g., promo engine) with impact analysis and change control** | **JIRA, Confluence, Azure Repos** | **Controlled architecture drift** |
| Requirements Management (Central Phase) | **Tracked functional + NFRs, linked them to architecture deliverables** | **JIRA, Confluence** | **Ensured traceability of all NFRs** |

| **TOGAF Phase** | **Key Strategy / Activity** | **Tools Used** | **Output / Goal** | **NFRs Addressed** |
| --- | --- | --- | --- | --- |
| **Preliminary** | **Identify stakeholders, define principles, (How architecture will be designed** **implemented, and governed.)**  **request for architecture work**  **Security by Design**  **Cloud-First:**  **Automation and Efficiency**  **Data Sovereignty** | **Confluence, Excel** | **Architecture Work Request** | **–** |
| **A. Architecture Vision** | **Define high-level architecture vision and NFR goals (scalability, HA, etc.)** | **Draw.io, Confluence** | **Vision Doc, Stakeholder Buy-in** | **Scalability, Security, RTO/RPO** |
| **B. Business Architecture** | **Model business domains, workflows (DDD - Product, Cart, Order, etc.)** | **Lucidchart, Confluence** | **Business Process Models, Domain Maps** | **Business Alignment** |
| **C. Information Systems Architecture** | **Define APIs, integrations, logical data models** | **Visual Studio, Swagger, Confluence** | **Service Contracts, Data Models** | **Secure APIs, Contract-first** |
| **D. Technology Architecture** | **Select Azure stack: AKS, Cosmos DB, AD B2C, Front Door, etc.** | **Azure Portal, Bicep, Azure Docs** | **Infra Architecture, Tech Mapping** | **HA/DR, Scalability, Security** |
| **E. Opportunities & Solutions** | **Identify reusable components, optimize cost with Azure-native services** | **Azure Calculator, Excel, Confluence** | **Solution Options, Gap Closure** | **Faster delivery, Cost-optimized** |
| **F. Migration Planning** | **Define phased rollout (e.g., Product → Cart → Payment)** | **Azure Boards, Confluence** | **Rollout Plan, Blue-Green Strategy** | **Zero downtime, DR readiness** |
| **G. Implementation Governance** | **Ensure compliance, code quality, pipeline gates** | **Azure DevOps, SonarQube, OWASP ZAP** | **Compliance Checklist, DevSecOps setup** | **Security, Quality, Governance** |
| **H. Architecture Change Management** | **Manage change requests, impact analysis, versioning** | **JIRA, Azure Repos, Confluence** | **Approved changes, Updated Docs** | **Change agility, Architecture Control** |
| **Requirements Management (Cross-cutting)** | **Map, trace, and validate functional & NFRs throughout lifecycle** | **JIRA, Confluence** | **NFR Mapping, Traceability Matrix** | **All NFRs maintained and validated** |

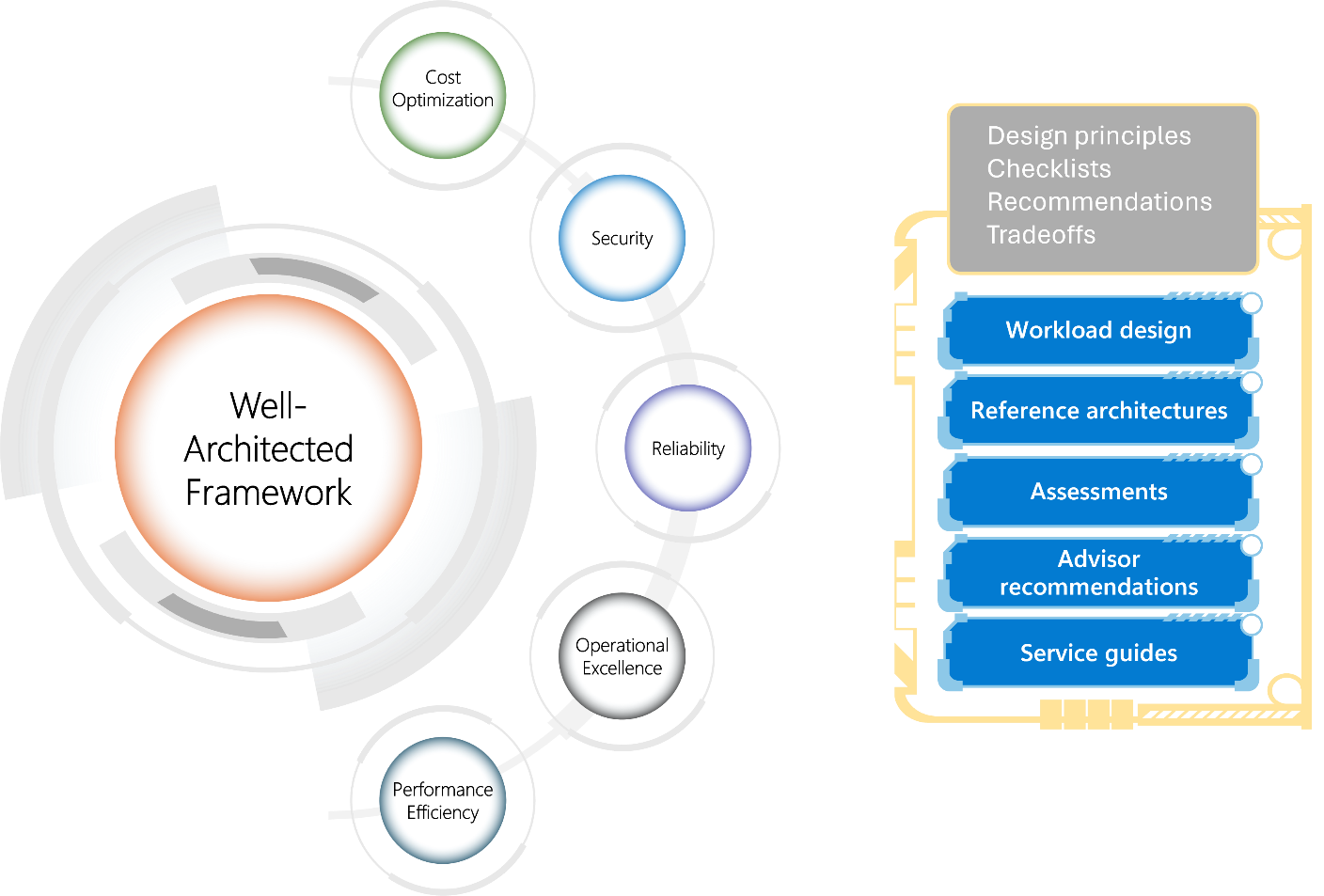
**🔐 NFR Strategy Recap**

| **NFR** | **How You Handled It** |
| --- | --- |
| **Scalability** | AKS (HPA), Azure Functions, Cosmos DB autoscale |
| **Security** | Azure AD B2C, OAuth2, API Gateway + WAF, Key Vault |
| **Resilience** | Retry logic, Circuit Breakers, Azure Service Bus |
| **Observability** | Azure Monitor, App Insights, Log Analytics |
| **RTO/RPO** | 15min/5min via Geo-redundant Storage, PITR, Azure SQL Auto Failover |
| **HA/DR** | Azure Front Door, Multi-region AKS & DB, SQL Always On |

**🎯 Customer Communication**

| **Activity** | **Tool Used** | **Why** |
| --- | --- | --- |
| Stakeholder Review | **Confluence** | For versioned documentation and approvals |
| Visual Presentation | **Draw.io / Visio** | Easy understanding of architecture |
| Roadmap Planning | **Azure Boards, JIRA** | Sprint-based delivery tracking |
| Governance Reviews | **Azure DevOps, SonarQube** | Maintain standards and track changes |

**Azure well architecture Framework ( Reliability, Security, Cost Optimization, Operation, Performance)**



**✅ Azure Well-Architected Framework – Interview Table**

| **Pillar** | **Description** | **Your Project Example / Azure Tool** |
| --- | --- | --- |
| **1. Reliability** | Ensure availability & recoverability | ✅ Azure SQL Always On ✅ Cosmos DB multi-region ✅ App Service Deployment Slots ✅ Azure Front Door with Health Probes |
| **2. Security** | Protect data & control access | 🔐 Azure AD B2C + RBAC 🔐 Azure Key Vault 🔐 HTTPS with custom certs 🔐 WAF + API Management with JWT |
| **3. Cost Optimization** | Maximize value & reduce waste | 💰 Auto-scale App Services 💰 Serverless Azure Functions 💰 Cosmos DB RU tuning + TTL 💰 Azure Budget Alerts 💰 Dev/Test Auto Shutdown |
| **4. Operational Excellence** | Monitor, deploy & recover efficiently | ⚙️ Azure DevOps CI/CD (YAML) ⚙️ Infra as Code (Bicep/ARM) ⚙️ Application Insights + Log Analytics ⚙️ Alerts + Dashboards |
| **5. Performance Efficiency** | Maximize performance under load | 🚀 Redis for caching 🚀 Cosmos DB with partition keys 🚀 gRPC for internal API calls 🚀 CDN for static files 🚀 Auto-Scale rules for APIs (HPA) |

**✅ Azure Well-Architected Framework – Complete Design Checklist**

| **Category** | **What It Means / Purpose** | **Tools / Best Practice Used in Project Example** |
| --- | --- | --- |
| **1. Design Principles** | Core philosophies to follow for any Azure architecture` | ✅ Design for failure ✅ Automate everything ✅ Least privilege access ✅ Test in production-like environments |
| **2. Workload Design** | Workload-specific architecture based on WAF pillars | 🧱 Identify workload type (e.g. Web app, API, ML) 🧱 Classify: Critical, HA, Geo-redundant |
| **3. Reference Architectures** | Azure-validated blueprints for common scenarios | 📘 Use official Azure Architecture Center 📘 Pick templates (e.g. microservices, event-driven, hybrid cloud) |
| **4. WAF Assessment** | Structured evaluation of how well your architecture aligns | 📊 Use [Azure Well-Architected Review Tool](https://learn.microsoft.com/en-us/assessments/) |
| **5. Azure Advisor** | Azure-native real-time recommendations | 🧠 Suggestions on cost, performance, security, reliability 🧠 Integrated into Azure Portal |
| **6. Azure Architecture Center** | Central repository of design patterns and use cases | 📚 Explore patterns: CQRS, Event Sourcing, Microservices 📚 Sample workloads (AKS, APIM, Azure Functions) |
| **7. Service Guides** | In-depth guides for each Azure service to align with WAF | 📄 E.g., App Service Guide, Cosmos DB Guide, AKS Guide – includes reliability, cost, scaling advice |
| **8. Implementation Tooling** | IaC + DevOps to apply WAF consistently | 🛠️ Terraform / Bicep / ARM 🛠️ Azure DevOps Pipelines / GitHub Actions 🛠️ Azure Monitor + Policy |
| **9. Governance & Policy** | Define rules to enforce best practices | 🛡️ Azure Policy 🛡️ Azure Blueprints 🛡️ Cost Management alerts 🛡️ Tagging, Naming standards |
| **🔟 Monitoring & Feedback** | Ensure visibility + continuous improvement | 📈 Azure Monitor 📈 Application Insights 📈 Log Analytics 📈 Alerts & Dashboards (Grafana, Workbook) |

| **🧩 NFR Area** | **💡 Design Considerations & Implementation (Azure + .NET)** |
| --- | --- |
| **1. Zero Downtime** | 1. Blue-Green Deployment (App Service/Azure AKS slots)  2. Azure Front Door with multi-region failover  3. Azure SQL Always On or Geo-replication  4. Cosmos DB with multi-region writes  5. Azure Traffic Manager (latency-based routing)  6. Canary deployments with feature flags  7. Health probes for AKS (readiness/liveness)  8. Low DNS TTL for fast failover  9. Azure Backup & automated DR testing  10. Rolling upgrades to avoid full downtime |
| **2. Scalability** | 1. AKS with HPA (Horizontal Pod Autoscaler)  2. Azure Functions for burst scale  3. Cosmos DB partitioning with proper PK strategy  4. Redis Cache for low-latency access  5. Event-driven patterns using Event Grid + Service Bus  6. Use Queue-based load leveling  7. Caching layers for catalogs, product info  8. Stateless microservices for scale-out  9. Auto-scale rules based on CPU/memory/queue depth  10. CDN for static content (images, scripts) |
| **3. Security** | 1. Azure AD B2C for customer identity federation  2. OAuth2 / OpenID Connect for secure auth  3. Internal microservices auth with IdentityServer4  4. Secure APIs with JWT, HTTPS, and scopes  5. Use Azure WAF + APIM for external APIs  6. Secrets in Azure Key Vault + RBAC  7. Use Managed Identity for secure resource access  8. Enable DDoS protection on public endpoints  9. Secure database access via private endpoints  10. Use Defender for Cloud for threat detection |
| **4. Cost Optimization** | 1. Azure Functions for batch/serverless logic  2. Cosmos DB RU/s control and TTL on docs  3. Auto-shutdown dev/test VMs and AKS  4. Spot VMs for batch workloads  5. Monitor with Azure Cost Management + Budgets  6. Choose B-series burstable VMs where applicable  7. Archive rarely-used data to Azure Archive Storage  8. Use Azure Reservations for committed workloads  9. Containerize apps to avoid VM over-provisioning  10. Optimize queries and reduce over-fetching |
| **5. Resilience** | 1. Outbox Pattern for message durability  2. Retry & Circuit Breaker using Polly or Dapr  3. Dead-letter queues (DLQ) for fault isolation  4. Cosmos DB multi-region fallback  5. Azure Front Door with regional failover  6. Durable Functions for long-running workflows  7. Service Bus with duplicate detection + sessions  8. Transaction log backups for SQL-based rollback  9. Load balancing with zone redundancy  10. Message replay via persisted event store |
| **6. Observability** | 1. Azure Monitor + Application Insights  2. Log Analytics for query-based dashboards  3. Distributed Tracing with OpenTelemetry & ActivitySource  4. App Map to visualize dependencies  5. Real-time alerts on CPU, latency, failure rates  6. Track custom metrics (business KPIs)  7. Enable diagnostic logs on APIM, App Gateway  8. Use Workbooks for unified dashboards  9. Enable container insights for AKS  10. Alert routing to Teams, Email, PagerDuty |
| **7. Operational Excellence** | 1. CI/CD using Azure DevOps or GitHub Actions  2. IaC with Bicep or Terraform  3. Feature Flags using App Config or LaunchDarkly  4. Deployment slots for safe rollback  5. Azure Policy for tagging, resource compliance  6. RBAC + PIM for least privilege access  7. Code scanning with SonarQube + Defender  8. Pipelines with gates for quality/security  9. Logging deployment history in pipelines  10. Self-healing checks with AKS probes |
| **8. Performance** | 1. Azure Redis Cache for product lookup, pricing  2. Cosmos DB indexing and partition tuning  3. SQL Server stored procs, filtered indexes  4. API response pagination + lazy loading  5. gRPC + HTTP/2 for fast inter-service calls  6. Data compression (GZip, Brotli)  7. Use CDN for static asset delivery  8. Async communication with queues  9. Profile APIs with App Insights + Profiler  10. Optimize GC settings and pool connections |
| **9. RTO / RPO** | 1. SQL Always On with Auto-failover groups  2. Cosmos DB Geo-replication with PITR  3. GRS-enabled Azure Storage backups  4. Azure Backup Vault + Recovery Services  5. Region-paired deployment strategy  6. Automated backup & restore pipelines  7. Test failovers quarterly  8. Set Recovery Plan in Azure Site Recovery  9. Log shipping & replication for legacy DBs  10. Document RTO/RPO in BCP Playbook |
| **10. Data Management** | 1. Cosmos DB Change Feed for projections/events  2. Data Sharding via customer/region partitioning  3. SQL Elastic Pools for cost-efficient scaling  4. Use PolyBase or ADF for data ingestion pipelines  5. Masking PII data for lower environments  6. Enable row-level security in SQL  7. Data versioning with soft deletes  8. Archive old orders/products into cold storage  9. Implement GDPR-compliant retention policy  10. Monitor data drift and schema changes |
| **Disaster Recovery** | | **Scenario** | **Solution Strategy** | | --- | --- | | **Azure Region Failure** | Used **Azure SQL Always On / Geo-replication**, **Cosmos DB Multi-region write**, and **Azure Front Door** with **Traffic Manager failover** | | **Azure VM Crash or Corruption** | Set up **Azure Backup Vault**, **Snapshot-based restore**, and **auto-scale with VM Scale Set** | | **Codebase Loss or Corruption in Dev/Test** | Used **Git Cherry-Pick**, **Branch Recovery**, and **Revert Commits** strategies via Azure Repos / GitHub | | **SQL Server DB Corruption or Accidental Delete** | Enabled **Point-in-Time Restore**, **Geo-restore**, and **Long-term backup retention (LTR)** | | **Cosmos DB Region Outage or Consistency Issues** | Enabled **Multi-region write** and **Automatic Failover Policy**, verified using **Consistency Levels (Session/Strong)** | | **App Configuration or Secrets Loss** | Integrated with **Azure Key Vault** backup + **ARM templates** for redeployment | | **Container or AKS Node Failure** | Configured **Multi-zone AKS cluster**, used **PodDisruptionBudgets**, and implemented **horizontal pod autoscaler** | | **Web App Down (App Service)** | Configured **Deployment Slots**, used **Swap with Preview**, enabled **auto-heal and backup/restore** | | **Service Bus Failure or Message Loss** | Enabled **Geo-redundant namespaces**, used **dead-letter queue**, and **replay from message archive (Event Hub Capture)** | | **DNS or Endpoint Failure** | Integrated with **Azure Front Door or Traffic Manager** with multi-region endpoints for **automatic failover and health probes** | |