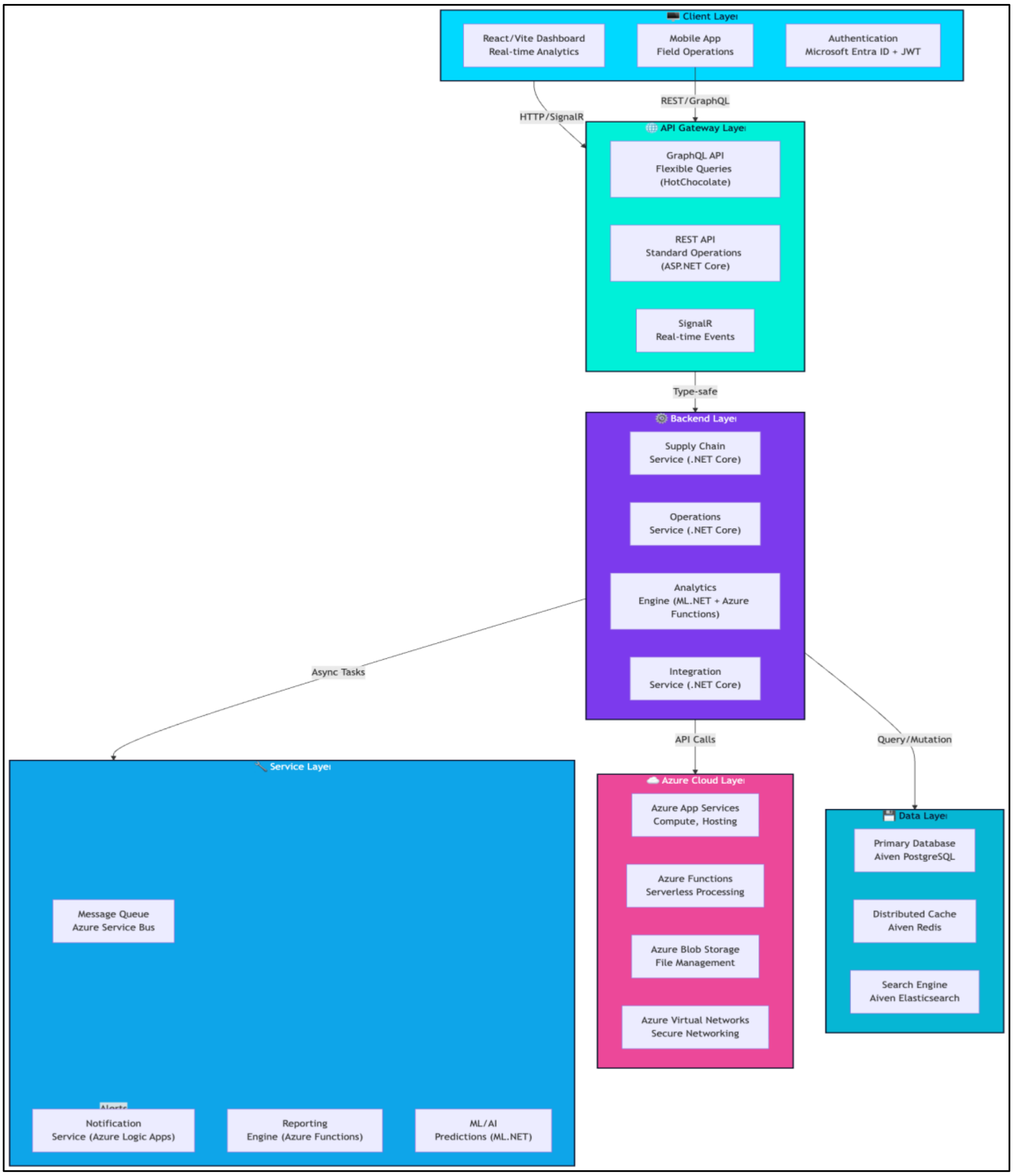
IMSOP - System Architecture

Overview

IMSOP (Intelligent Multi-Cloud Supply Chain & Operations Platform) is an enterprise-grade supply chain management and operations platform. It provides comprehensive visibility, control, and optimization across Azure-centric environments with optional multi-cloud support via Azure Arc, ensuring robustness, security, and high performance through containerization, IaC, and advanced monitoring.

System Architecture Diagram



Component Details

Client Layer

- **React/Vite Dashboard**: Comprehensive supply chain analytics and management, deployed on Vercel for fast static hosting with SSR support.
- **Mobile App**: Field operations and real-time updates (React Native integration).
- **Authentication**: Microsoft Entra ID (Azure AD) with JWT tokens and OAuth 2.0.

API Gateway Layer

- **GraphQL API**: Flexible query language using HotChocolate for complex data requirements.
- **REST API**: Standard CRUD operations with ASP.NET Core Web API, secured via Azure API Management.
- **SignalR**: Real-time event streaming and notifications for high-performance updates.

Backend Layer

- **Supply Chain Service**: Procurement, inventory, logistics management (DDD with SOLID principles).
- **Operations Service**: Workflow automation, task management (Async processing).
- **Analytics Engine**: Predictive analytics using ML.NET and Azure Functions.
- **Integration Service**: Third-party API integrations via Azure Logic Apps and Microsoft Graph API.

Data Layer

- **Aiven PostgreSQL Database**: Managed primary data storage with ACID compliance and auto-scaling.
- **Aiven Redis Cache**: High-performance caching layer for distributed sessions and data.
- **Aiven Elasticsearch**: Full-text search and log aggregation, integrated with Azure Monitor.

Azure Cloud Layer

- **Azure App Services**: Hosting for .NET Core microservices.
- **Azure Functions**: Serverless compute for event-driven tasks.
- **Azure Blob Storage**: Secure file storage with encryption.
- **Azure Virtual Networks**: Isolated networking with RBAC and Managed Identities.

Service Layer

- **Message Queue**: Azure Service Bus for asynchronous task processing.
- **Notification Service**: Azure Logic Apps for email/SMS/push notifications.
- **Reporting Engine**: Azure Functions for PDF generation and scheduled reports.
- **ML/AI**: ML.NET models deployed in Azure Functions for predictions.

Data Flow

Supply Chain Order Flow

1. Intake & Validation

The process begins when a user submits an order. Before any data hits the database, it must pass a logic gate.

- **Order Creation:** The initial POST request.
- **Validation (FluentValidation):** Ensures the data is structurally sound (e.g., valid email formats, non-empty fields, positive quantities).
- **Inventory Check (EF Core):** A synchronous check against the database to ensure the items are currently in stock.

2. Decoupling & Queuing

Once validated, the system avoids "blocking" the user by offloading the heavy lifting.

- **Queue Processing (Azure Service Bus):** The order is published as a message to a topic or queue. This ensures that even if the supplier service is down, the order is safely persisted and ready for processing.

3. Execution & Fulfillment

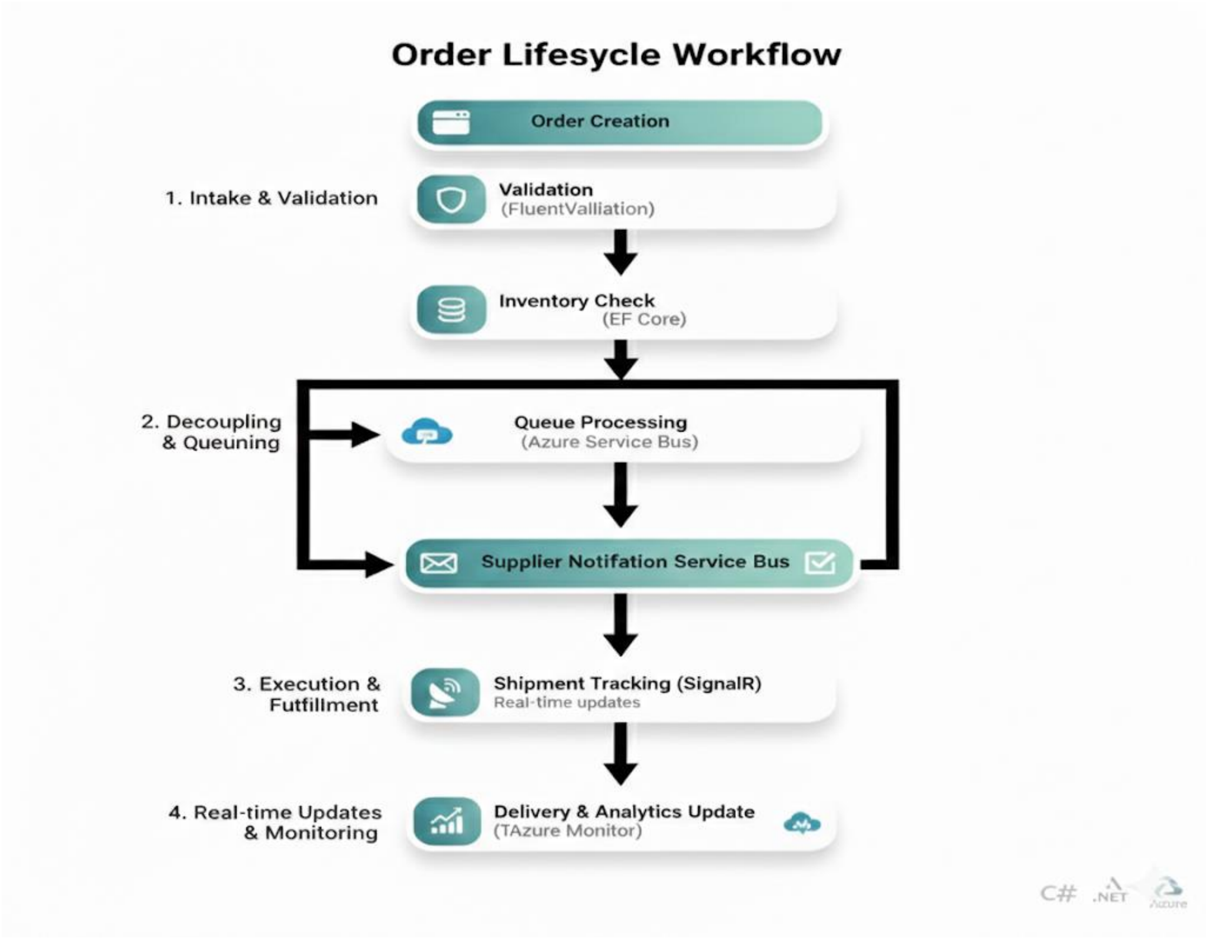
The backend workers pick up the message from the Service Bus to complete the transaction.

- **Supplier Notification:** An automated trigger (likely a Function or Microservice) notifies the warehouse or third-party supplier.
- **Fulfillment:** The physical process of picking, packing, and readying the item for shipping.

4. Real-time Updates & Monitoring

As the order moves toward the customer, the system provides transparency and logs performance.

- **Shipment Tracking (SignalR):** Instead of the user refreshing their page, SignalR pushes "Live" status updates (e.g., "Picked," "Shipped") directly to the client UI in real-time.
- **Delivery & Analytics Update (Azure Monitor):** Once the cycle is complete, the telemetry data is sent to Azure Monitor/Application Insights to track success rates, latency, and potential bottlenecks.



Data Pipeline Architecture

1. Ingestion & Pre-processing

The journey begins with the raw data entry point.

- **Data Sources:** External APIs, IoT sensors, or database logs.
- **Collection (.NET Worker):** A lightweight, long-running background service responsible for polling or listening to data sources and pushing them into the cloud environment.

2. Transformation & Storage

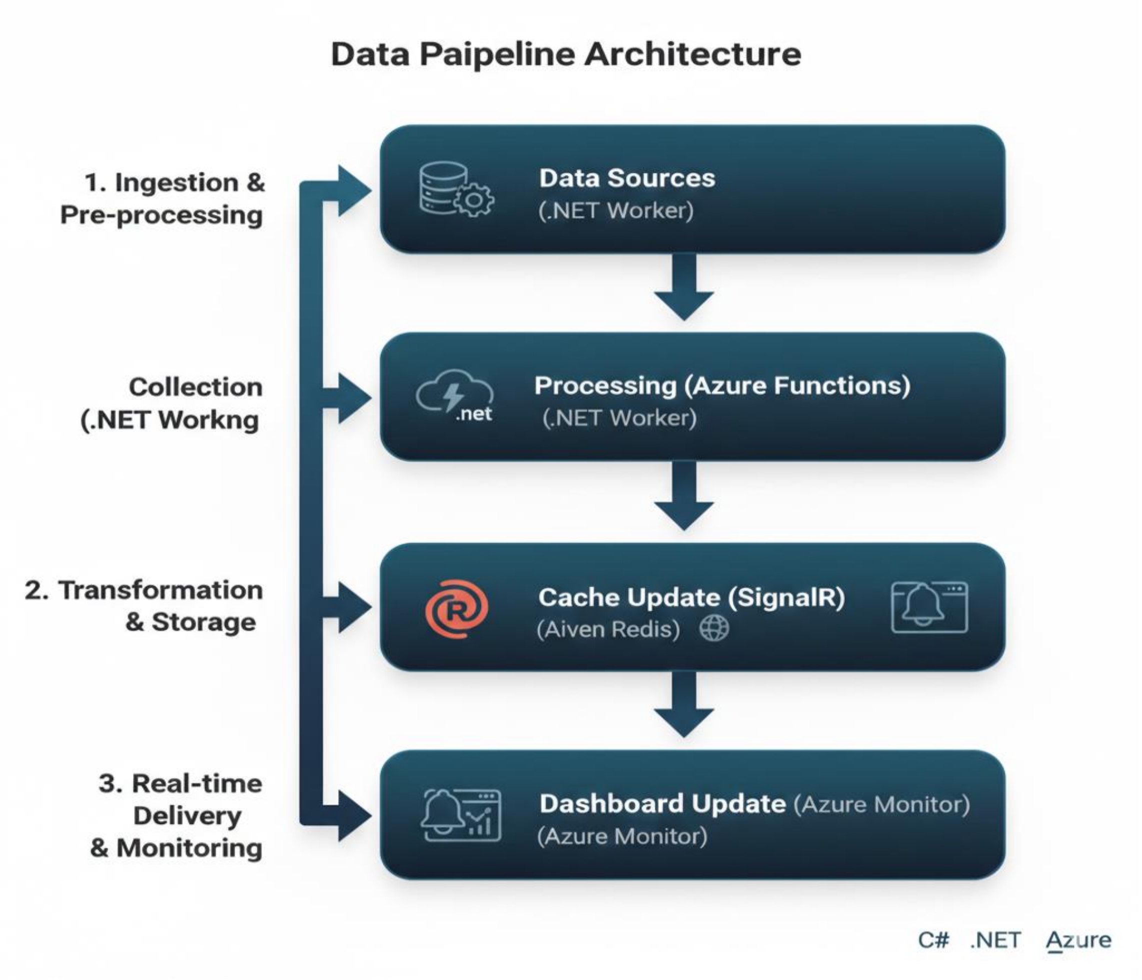
Once the data is inside the cloud, it needs to be "cleaned" and stored for fast access.

- **Processing (Azure Functions):** Serverless compute handles the business logic, data normalization, and transformation. This scales automatically based on the volume of incoming data.
- **Cache Update (Aiven Redis):** Instead of hitting a slow primary database, processed data is pushed to a high-speed **Redis** instance. This ensures the "latest state" of the data is available with sub-millisecond latency.

3. Real-time Delivery & Monitoring

The final layer focuses on getting that data in front of the right eyes immediately.

- **Dashboard Update (SignalR):** Rather than having users refresh a browser, SignalR pushes the updated Redis values directly to the front-end dashboard via WebSockets.
- **Alert Generation (Azure Monitor):** If the data processing identifies an anomaly or exceeds a predefined threshold, Azure Monitor triggers automated alerts (Email, SMS, or Webhooks).



Data Paipeline Architecture
1. Ingestion & Pre-processing
Collection (.NET Workng
2. Transformation & Storage
3. Real-time Delivery & Monitoring
Data Sources
(.NET Worker)
.net
Processing (Azure Functions)
(.NET Worker)
Cache Update (SignalR)
(Aiven Redis)
Dashboard Update (Azure Monitor)
(Azure Monitor)
C#
.NET
Azure

Hybrid Request & Integration Workflow

1. The Gateway Layer

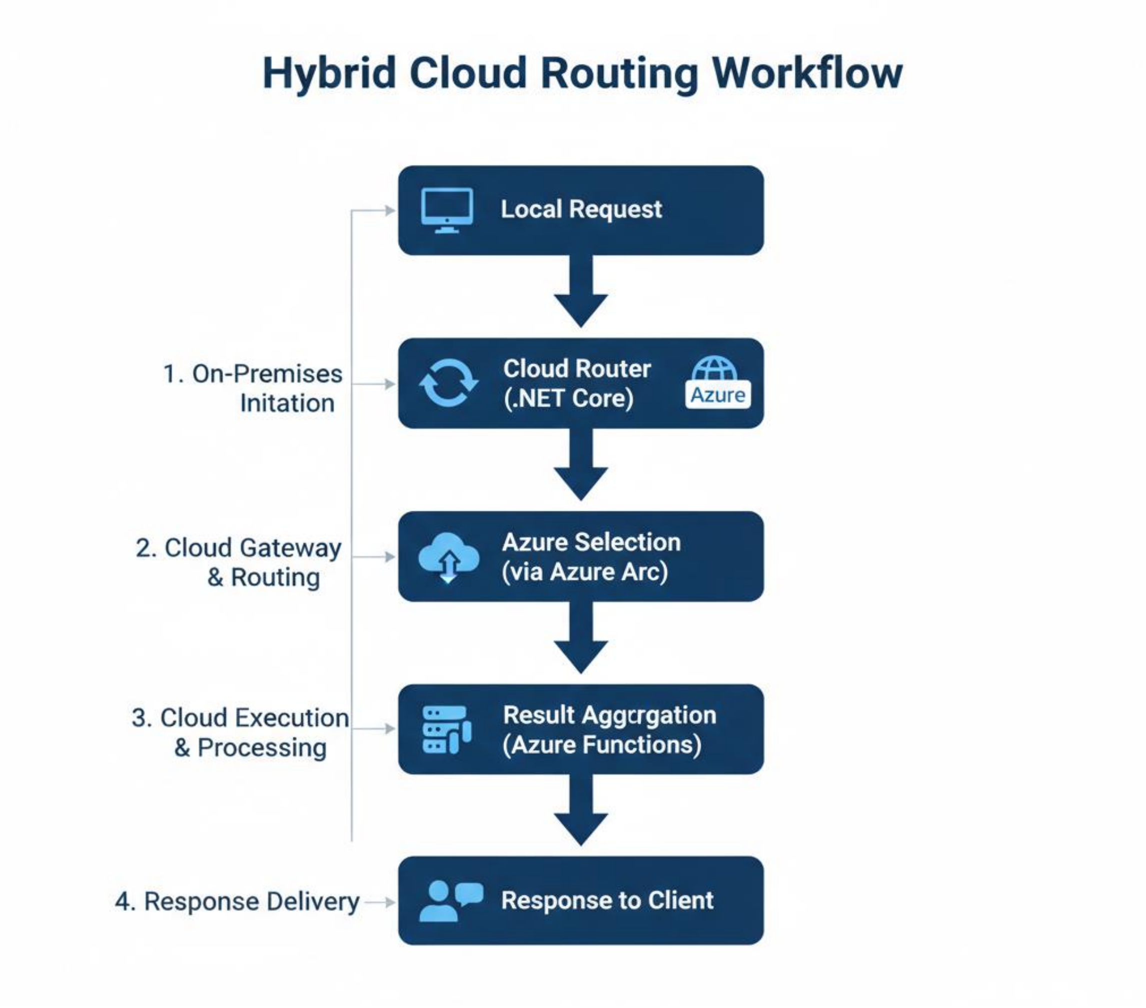
- **Local Request:** This is the entry point, likely originating from an on-premises application or a local user terminal.
- **Cloud Router (.NET Core):** A high-performance middleware built on .NET Core that acts as the intelligent traffic cop. It determines how to handle the request based on geography, load, or security rules.

2. The Hybrid Bridge

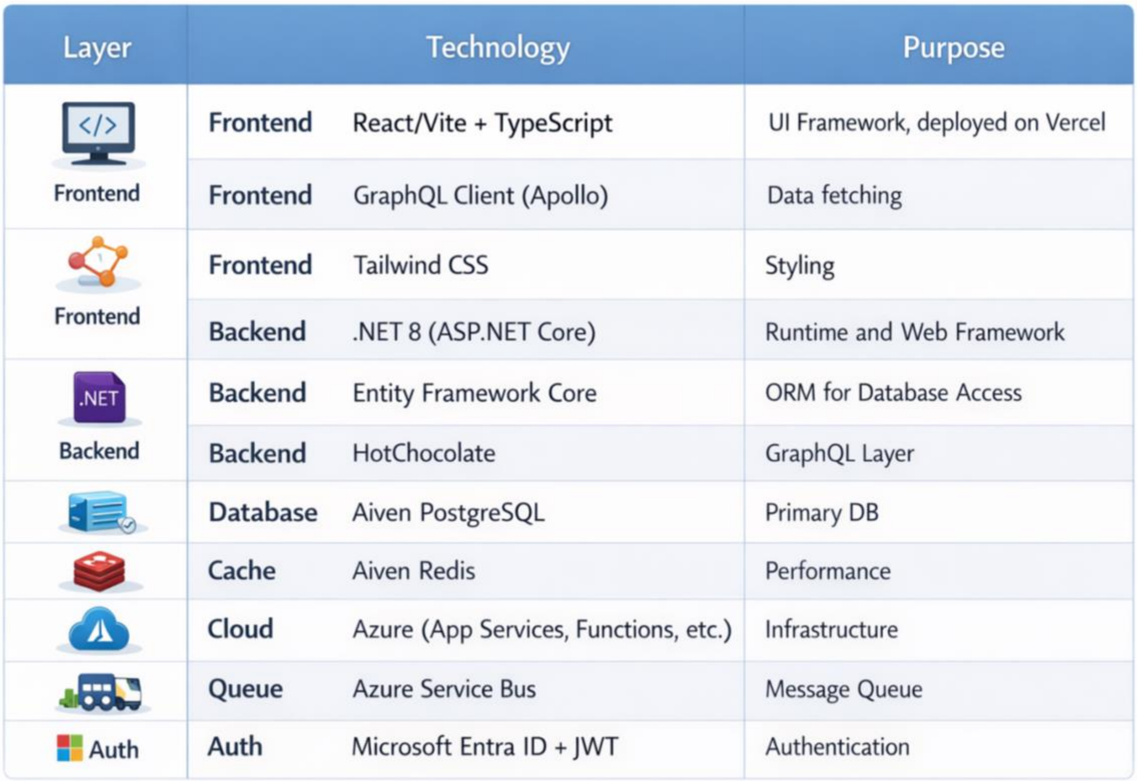
- **Azure Selection (via Azure Arc):** This is the "brain" of the hybrid setup. **Azure Arc** allows you to manage non-Azure resources as if they were native. The router uses Arc to select the most appropriate Azure region or localized resource to handle the heavy lifting.
- **Azure API Call (SDK):** Once the target is identified, the system uses the official Azure SDK to securely trigger cloud services (like Storage, AI, or Compute).

3. Aggregation & Delivery

- **Result Aggregation (Azure Functions):** Since an API call might return raw or fragmented data, a serverless **Azure Function** sits in the middle. It "massages" the data, combines it with other necessary context, and prepares the final payload.
- **Response to Client:** The cleaned, aggregated data is sent back through the router to the local user, completing the cycle with minimal latency.


Hybrid Cloud Routing Workflow
Local Request
1. On-Premises Initation
Cloud Router (.NET Core)
Azure
2. Cloud Gateway & Routing
Azure Selection (via Azure Arc)
3. Cloud Execution & Processing
Result Aggcrgation (Azure Functions)
4. Response Delivery
Response to Client

Technology Stack



Layer	Technology		Purpose
Frontend	Frontend	React/Vite + TypeScript	UI Framework, deployed on Vercel
	Frontend	GraphQL Client (Apollo)	Data fetching
Frontend	Frontend	Tailwind CSS	Styling
	Backend	.NET 8 (ASP.NET Core)	Runtime and Web Framework
Backend	Backend	Entity Framework Core	ORM for Database Access
	Backend	HotChocolate	GraphQL Layer
	Database	Aiven PostgreSQL	Primary DB
	Cache	Aiven Redis	Performance
	Cloud	Azure (App Services, Functions, etc.)	Infrastructure
	Queue	Azure Service Bus	Message Queue
Auth	Auth	Microsoft Entra ID + JWT	Authentication

Key Features

1. Supply Chain Management

- Procurement automation
- Inventory optimization
- Supplier management
- Purchase order tracking

2. Operations Management

- Workflow automation
- Task management
- Resource allocation
- Performance tracking

3. Analytics & Insights

- Real-time dashboards
- Predictive analytics
- Anomaly detection
- Custom reports

4. Azure-Centric Support

- Azure App Services integration
- Azure Functions for serverless
- Azure Arc for hybrid/multi-cloud
- Secure networking with Virtual Networks

5. Integration Capabilities

- ERP system integration via Azure Logic Apps
- Third-party API support (OAuth 2.0)
- Data synchronization with Microsoft Graph API
- Webhook support

Security Architecture

Authentication

- Microsoft Entra ID for third-party integrations
- JWT for API authentication
- Multi-factor authentication support
- Session management with Managed Identities

Authorization

- Role-based access control (RBAC) via Azure
- Attribute-based access control (ABAC)
- Resource-level permissions
- Audit logging with Azure Monitor

Data Protection

- End-to-end encryption (Azure Key Vault)
- Database encryption at rest (Aiven)
- TLS/SSL in transit
- Data anonymization and secrets management

Scalability Considerations

Horizontal Scaling

- Stateless microservices in Docker/Kubernetes
- Load balancing via Azure App Service
- Database replication (Aiven auto-scaling)
- Cache distribution (Aiven Redis Cluster)

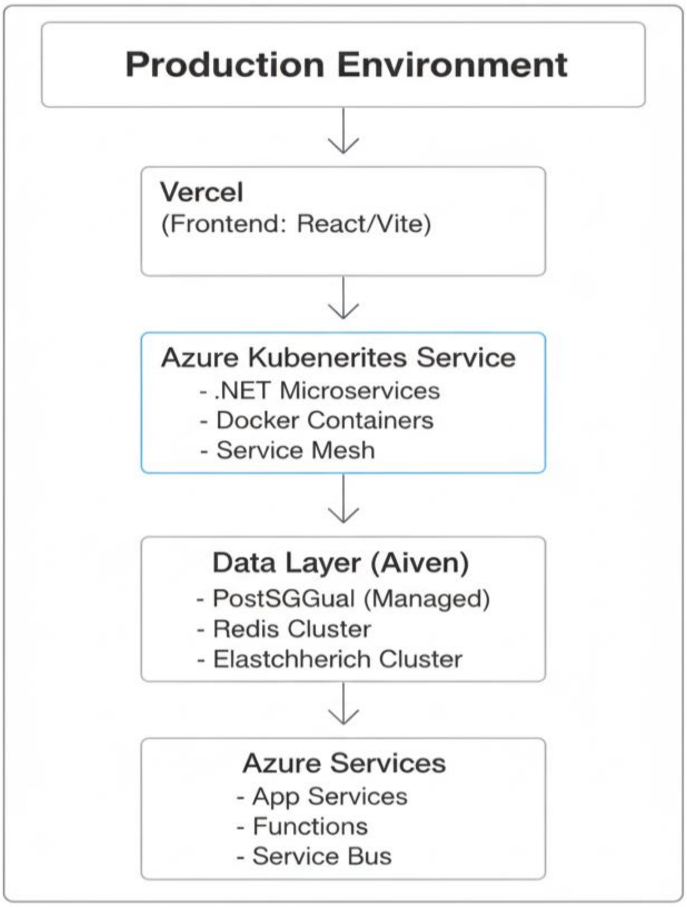
Performance Optimization

- Query optimization with EF Core
- Caching strategies (IDistributedCache)
- Batch processing in Azure Functions
- Asynchronous operations with async/await in .NET

Monitoring & Observability

- Azure Monitor and Application Insights for logging
- Log Analytics for centralized logs
- Alerting and performance optimization
- Root cause analysis with troubleshooting tools

Deployment Architecture



Deployment Tools:

- IaC: Azure Bicep/ARM Templates for infrastructure provisioning.
- CI/CD: Azure DevOps Pipelines or GitHub Actions for automated builds, tests (TDD), and deployments.
- Static Assets: GitHub Pages for documentation/hosting static parts if needed, Render for backend preview environments.

SOLID Principles Implementation

Single Responsibility

- Each service handles one domain
- Clear separation of concerns
- Focused business logic

Open/Closed

- Extensible through plugins
- New integrations without modification
- Interface-based design

Liskov Substitution

- Consistent service interfaces
- Predictable behavior
- Type-safe operations

Interface Segregation

- Minimal required dependencies
- Focused service contracts
- Specific API endpoints

Dependency Inversion

- Services depend on abstractions
- Dependency injection pattern
- Plugin architecture

Performance Metrics

- **API Response Time**: < 200ms (p95)
- **GraphQL Query Time**: < 500ms (p95)
- **Real-time Event Latency**: < 100ms
- **Dashboard Load Time**: < 2s
- **Database Query Time**: < 50ms (p95)
- **Cache Hit Rate**: > 85%
- **System Availability**: > 99.9%

Future Enhancements

1. **Advanced Analytics**
   - Machine learning models
   - Predictive maintenance
   - Demand forecasting
2. **Blockchain Integration**
   - Supply chain transparency
   - Smart contracts
   - Immutable audit trail
3. **IoT Integration**
   - Real-time tracking
   - Sensor data collection
   - Automated alerts
4. **Advanced Automation**
   - RPA integration
   - Workflow optimization
   - Intelligent routing
5. **Sustainability**
   - Carbon footprint tracking
   - Green logistics optimization
   - ESG reporting