# Multi-Agent Azure Data Analytics Platform Setup

## Project Directory Structure

```
azure-data-platform/
├── .azure-devops/
│   ├── pipelines/
│   │   ├── infrastructure-deploy.yml
│   │   ├── databricks-deploy.yml
│   │   ├── api-deploy.yml
│   │   └── powerbi-deploy.yml
│   └── templates/
│       ├── infrastructure.yml
│       ├── databricks.yml
│       ├── dotnet-api.yml
│       └── powerbi.yml
├── infrastructure/
│   ├── bicep/
│   │   ├── main.bicep
│   │   ├── modules/
│   │   │   ├── storage.bicep
│   │   │   ├── databricks.bicep
│   │   │   ├── data-factory.bicep
│   │   │   ├── app-service.bicep
│   │   │   ├── key-vault.bicep
│   │   │   └── monitoring.bicep
│   │   └── environments/
│   │       ├── dev.bicepparam
│   │       ├── staging.bicepparam
│   │       └── prod.bicepparam
│   └── scripts/
│       ├── deploy.ps1
│       └── setup-rbac.ps1
├── src/
│   ├── databricks/
│   │   ├── notebooks/
│   │   │   ├── bronze-ingestion/
│   │   │   │   ├── ingest-sales-data.py
│   │   │   │   ├── ingest-customer-data.py
│   │   │   │   └── ingest-iot-stream.py
│   │   │   ├── silver-transformation/
│   │   │   │   ├── clean-sales-data.py
│   │   │   │   ├── clean-customer-data.py
│   │   │   │   └── clean-device-events.py
│   │   │   └── gold-aggregation/
│   │   │       ├── create-dimensions.py
│   │   │       ├── create-facts.py
│   │   │       └── calculate-kpis.py
│   │   ├── jobs/
│   │   │   ├── bronze-to-silver-job.json
│   │   │   ├── silver-to-gold-job.json
│   │   │   └── full-pipeline-job.json
│   │   └── tests/
│   │       ├── test_bronze_ingestion.py
│   │       ├── test_silver_transformation.py
│   │       └── test_gold_aggregation.py
│   ├── api/
│   │   ├── DataPlatform.API/
│   │   │   ├── Controllers/
│   │   │   │   ├── CustomersController.cs
│   │   │   │   ├── SalesController.cs
│   │   │   │   └── DevicesController.cs
│   │   │   ├── Models/
│   │   │   │   ├── Customer.cs
│   │   │   │   ├── Sales.cs
│   │   │   │   └── Device.cs
│   │   │   ├── Services/
│   │   │   │   ├── IDataService.cs
│   │   │   │   ├── DatabricksDataService.cs
│   │   │   │   └── CacheService.cs
│   │   │   ├── Configuration/
│   │   │   │   ├── AppSettings.cs
│   │   │   │   └── DatabaseSettings.cs
│   │   │   ├── Program.cs
│   │   │   └── DataPlatform.API.csproj
│   │   └── DataPlatform.Tests/
│   │       ├── Controllers/
│   │       ├── Services/
│   │       ├── Integration/
│   │       └── DataPlatform.Tests.csproj
│   └── powerbi/
│       ├── datasets/
│       │   ├── customer-360.pbit
│       │   ├── device-health.pbit
│       │   └── revenue-analytics.pbit
│       ├── reports/
│       ├── scripts/
│       │   ├── deploy-datasets.ps1
│       │   └── refresh-datasets.ps1
│       └── tests/
├── data/
│   ├── samples/
│   │   ├── sales_orders.csv
│   │   ├── customers.csv
│   │   ├── devices.csv
│   │   └── device_events.json
│   └── schema/
│       ├── bronze-schema.json
│       ├── silver-schema.json
│       └── gold-schema.json
├── docs/
│   ├── architecture.md
│   ├── deployment-guide.md
│   ├── api-documentation.md
│   └── troubleshooting.md
├── tests/
│   ├── end-to-end/
│   └── performance/
├── .gitignore
├── README.md
└── azure-pipelines.yml
```

## Agent Specializations & Context Files

### 1. Infrastructure Agent (Azure/DevOps)

**Responsibilities**: Bicep templates, Azure DevOps pipelines, RBAC setup **Context File**: `infrastructure/context.md`

### 2. Databricks Agent (Python/PySpark)

**Responsibilities**: ETL notebooks, job configurations, data transformations **Context File**: `src/databricks/context.md`

### 3. .NET API Agent (C#)

**Responsibilities**: REST API, authentication, data access layer **Context File**: `src/api/context.md`

### 4. Power BI Agent (Power BI/PowerShell)

**Responsibilities**: Datasets, reports, deployment scripts **Context File**: `src/powerbi/context.md`

---

## Agent Prompts

### 🏗️ Infrastructure Agent Prompt

**Role**: Azure Infrastructure & DevOps Specialist **Scope**: Infrastructure as Code, CI/CD Pipelines, Security Setup

#### Primary Tasks:

1. Create Bicep templates for all Azure resources
2. Set up Azure DevOps pipelines with proper environments (dev/staging/prod)
3. Configure Azure AD, RBAC, and Managed Identities
4. Implement Azure Key Vault for secrets management
5. Set up monitoring and alerting with Azure Monitor

**Context Requirements:**

- **READ FIRST**: `infrastructure/context.md` for naming conventions, resource configurations, and security requirements
- **Environment Strategy**: dev/staging/prod with parameter files
- **Security**: Least-privilege RBAC, Managed Identity for all services
- **Cost Optimization**: Use appropriate SKUs for each environment

**Deliverables:**

1. **Bicep Templates** (`infrastructure/bicep/`)
   - Main template with all resource modules
   - Separate modules for each service (storage, databricks, etc.)
   - Environment-specific parameter files
2. **Azure DevOps Pipelines** (`.azure-devops/pipelines/`)
   - Infrastructure deployment pipeline
   - Environment-specific variable groups
   - Approval gates for production
3. **RBAC Configuration** (`infrastructure/scripts/`)
   - PowerShell scripts for role assignments
   - Service principal setup for automation

**Quality Gates:**

- All resources must use Managed Identity
- Secrets stored only in Key Vault
- Resource names follow naming convention
- Cost optimization features enabled
- Monitoring and alerting configured

**Dependencies:**

- Must complete before other agents can deploy
- Provides connection strings and resource names to other agents

---

## 🔄 Databricks Agent Prompt

**Role**: Data Engineering & ETL Specialist **Scope**: PySpark notebooks, Delta Lake, Data pipelines

### Primary Tasks:

1. Create bronze-to-silver-to-gold ETL notebooks
2. Implement data quality checks and schema validation
3. Set up streaming ingestion from IoT Hub
4. Configure Databricks jobs with proper scheduling
5. Create comprehensive test suites

### Context Requirements:

- **READ FIRST**: `src/databricks/context.md` for data schemas, transformation rules, and quality standards
- **Data Architecture**: Bronze (raw) → Silver (cleaned) → Gold (analytics-ready)
- **Framework**: PySpark with Delta Lake
- **Quality**: Data validation, deduplication, error handling

### Deliverables:

1. **ETL Notebooks** (`src/databricks/notebooks/`)
   - Bronze ingestion: CSV from Blob, JSON from IoT Hub
   - Silver transformation: cleaning, standardization, validation
   - Gold aggregation: star schema, KPI calculations
2. **Job Configurations** (`src/databricks/jobs/`)
   - JSON job definitions for Databricks workflows
   - Scheduling and dependency management
   - Error handling and retry logic
3. **Test Suite** (`src/databricks/tests/`)
   - pytest-based unit tests for each transformation
   - Data quality validation tests
   - Integration tests with sample data

### Quality Gates:

- All transformations handle schema drift
- Data quality checks on every layer
- Proper error logging and monitoring
- Delta Lake optimize and vacuum scheduled
- Tests achieve >90% coverage

### Dependencies:

- Requires infrastructure deployment first
- Provides data schemas to API agent
- Sample data for testing other components

---

## 🚀 .NET API Agent Prompt

**Role**: Backend API & Authentication Specialist
**Scope**: REST API, Azure AD integration, Data access layer

### Primary Tasks:

1. Build .NET 8 Web API with clean architecture
2. Implement Azure AD authentication with proper scopes
3. Create data access layer connecting to Gold layer tables
4. Add caching, logging, and performance monitoring
5. Comprehensive testing with xUnit

### Context Requirements:

- **READ FIRST**: `src/api/context.md` for API specifications, authentication requirements, and performance standards
- **Architecture**: Clean architecture with dependency injection
- **Security**: Azure AD OAuth2, role-based access control
- **Performance**: Response times <1sec, proper caching strategy

### Deliverables:

1. **Web API Project** (`src/api/DataPlatform.API/`)
   - Controllers for Customers, Sales, Devices endpoints
   - Models matching Gold layer schema
   - Services for data access and business logic
   - Configuration for multiple environments
2. **Authentication & Authorization**
   - Azure AD integration with proper scopes
   - JWT token validation
   - Role-based access control
3. **Testing Projects** (`src/api/DataPlatform.Tests/`)
   - Unit tests for controllers and services
   - Integration tests with test database
   - Performance tests for API endpoints

### Quality Gates:

- All endpoints properly authenticated
- Response times under 1 second
- Proper error handling and logging
- API documentation with Swagger
- Tests achieve >85% coverage
- Security scanning passes

### Dependencies:

- Requires infrastructure and Databricks completion
- Consumes Gold layer data schemas
- Provides API endpoints for Power BI integration

---

## 📊 Power BI Agent Prompt

**Role**: Business Intelligence & Visualization Specialist **Scope**: Power BI datasets, reports, automated deployment

### Primary Tasks:

1. Create Power BI datasets connected to Gold layer
2. Build interactive dashboards for Customer 360, Device Health, Revenue Analytics
3. Implement automated refresh and deployment
4. Create PowerShell scripts for CI/CD integration
5. Set up usage monitoring and performance optimization

### Context Requirements:

- **READ FIRST**: `src/powerbi/context.md` for dashboard requirements, data connections, and visualization standards
- **Data Sources**: Connect to Gold layer tables via direct query or import
- **Dashboards**: Customer 360, Device Health, Revenue vs Device Health
- **Deployment**: Automated via PowerShell and Power BI REST API

### Deliverables:

1. **Power BI Templates** (`src/powerbi/datasets/`)
   - .pbit template files for each dashboard
   - Proper data model relationships
   - Optimized DAX calculations
2. **Deployment Scripts** (`src/powerbi/scripts/`)
   - PowerShell scripts for automated deployment
   - Dataset refresh automation
   - Environment-specific configurations
3. **Documentation & Testing**
   - User guides for each dashboard
   - Performance optimization documentation
   - Automated testing of report functionality

### Quality Gates:

- Dashboards load in <5 seconds
- Proper security (RLS) implemented
- Mobile-responsive design
- Automated refresh working
- Usage analytics configured

### Dependencies:

- Requires Databricks Gold layer completion
- May integrate with API endpoints for real-time data
- Final component in the data pipeline

---

## Context Files for Each Agent

### Infrastructure Context (`infrastructure/context.md`)

```markdown
# Infrastructure Agent Context

## Naming Conventions
- Resource Group: `rg-dataplatform-{env}`
- Storage Account: `stdataplatform{env}{random}`
- Key Vault: `kv-dataplatform-{env}`
- Databricks: `dbw-dataplatform-{env}`

## Environment Configuration
- **dev**: Basic SKUs, single region, minimal redundancy
- **staging**: Standard SKUs, production-like setup
- **prod**: Premium SKUs, geo-redundancy, high availability

## Security Requirements
- All resources use Managed Identity
- Network security groups for all subnets
- Private endpoints for storage and Key Vault
- RBAC with least privilege principle

## Cost Optimization
- Auto-pause for dev Databricks clusters
- Lifecycle management for blob storage
- Reserved instances for production
```

### Databricks Context (`src/databricks/context.md`)

```markdown
# Databricks Agent Context

## Data Quality Rules
- Remove duplicates based on primary keys
- Validate required fields not null
- Standardize date formats to ISO 8601
- Currency conversion to USD
- Email validation and cleansing

## Schema Evolution
- Handle new columns gracefully
- Version all schema changes
- Maintain backward compatibility
- Document schema changes

## Performance Standards
- Bronze ingestion: <30min for daily batch
- Silver transformation: <15min per table
- Gold aggregation: <10min for all KPIs
- Streaming latency: <5min for IoT events

## Error Handling
- Retry failed jobs 3 times
- Dead letter queue for corrupted data
- Alerting on job failures
- Data lineage tracking
```

## API Context (`src/api/context.md`)

```markdown
# .NET API Agent Context

## API Specifications
- RESTful design with proper HTTP verbs
- JSON responses with camelCase
- Proper HTTP status codes
- Comprehensive error responses

## Authentication & Authorization
- Azure AD OAuth2 with scopes:
  - `api://dataplatform/read`: Read-only access
  - `api://dataplatform/write`: Write access
  - `api://dataplatform/admin`: Admin operations

## Performance Requirements
- Response time <1 second for 95% of requests
- Support 1000 concurrent users
- Implement response caching (5-minute TTL)
- Rate limiting: 100 requests/minute per user

## Data Access Patterns
- Use connection pooling
- Implement read replicas for queries
- Cache frequently accessed data
- Proper disposal of resources
```

### Power BI Context (`src/powerbi/context.md`)

```markdown
# Power BI Agent Context

## Dashboard Requirements

### Customer 360 Dashboard
- Customer profile with key metrics
- Sales history and trends
- Device status and health
- Support ticket summary

### Device Health Dashboard
- Real-time device status grid
- Battery health distribution
- Error code frequency analysis
- Predictive maintenance alerts

### Revenue vs Device Health
- Correlation between device issues and churn
- Revenue impact of device problems
- Customer satisfaction metrics

## Performance Standards
- Dashboard load time <5 seconds
- Refresh time <10 minutes
- Support 100+ concurrent users
- Mobile responsive design

## Security & Governance
- Row-level security by region/customer
- Certified datasets only
- Version control for reports
- Usage monitoring and optimization
```

---

## Handoff Protocols Between Agents

### 1. Infrastructure → All Agents

**Deliverables**: Resource names, connection strings, managed identity details **Format**: JSON configuration files in `/infrastructure/outputs/`

### 2. Databricks → API Agent

**Deliverables**: Gold layer table schemas, sample queries **Format**: SQL DDL files and JSON schema definitions

### 3. Databricks → Power BI Agent

**Deliverables**: Data model, table relationships, calculated columns **Format**: Power BI data model documentation

### 4. API Agent → Power BI Agent

**Deliverables**: API endpoint specifications, authentication details **Format**: OpenAPI/Swagger specifications

---

## Review Checklists for Each Agent

### Infrastructure Agent PR Checklist

- [ ] All Bicep templates validate successfully
- [ ] Environment parameters properly configured
- [ ] RBAC assignments follow least privilege
- [ ] Secrets stored only in Key Vault
- [ ] Cost optimization features enabled
- [ ] Monitoring and alerting configured
- [ ] Pipeline deployment tests pass

### Databricks Agent PR Checklist

- [ ] Data quality checks implemented
- [ ] Schema drift handling in place
- [ ] Error logging and monitoring added
- [ ] Unit tests cover all transformations
- [ ] Integration tests with sample data pass
- [ ] Performance benchmarks met
- [ ] Documentation updated

### .NET API Agent PR Checklist

- [ ] Authentication properly implemented
- [ ] All endpoints secured
- [ ] Response times under 1 second
- [ ] Proper error handling and logging
- [ ] Unit and integration tests pass
- [ ] API documentation generated
- [ ] Security scanning passes

## Power BI Agent PR Checklist

- [ ] Dashboards load under 5 seconds
- [ ] Data model optimized
- [ ] Row-level security implemented
- [ ] Mobile responsive design
- [ ] Automated deployment working
- [ ] User documentation complete
- [ ] Performance metrics captured