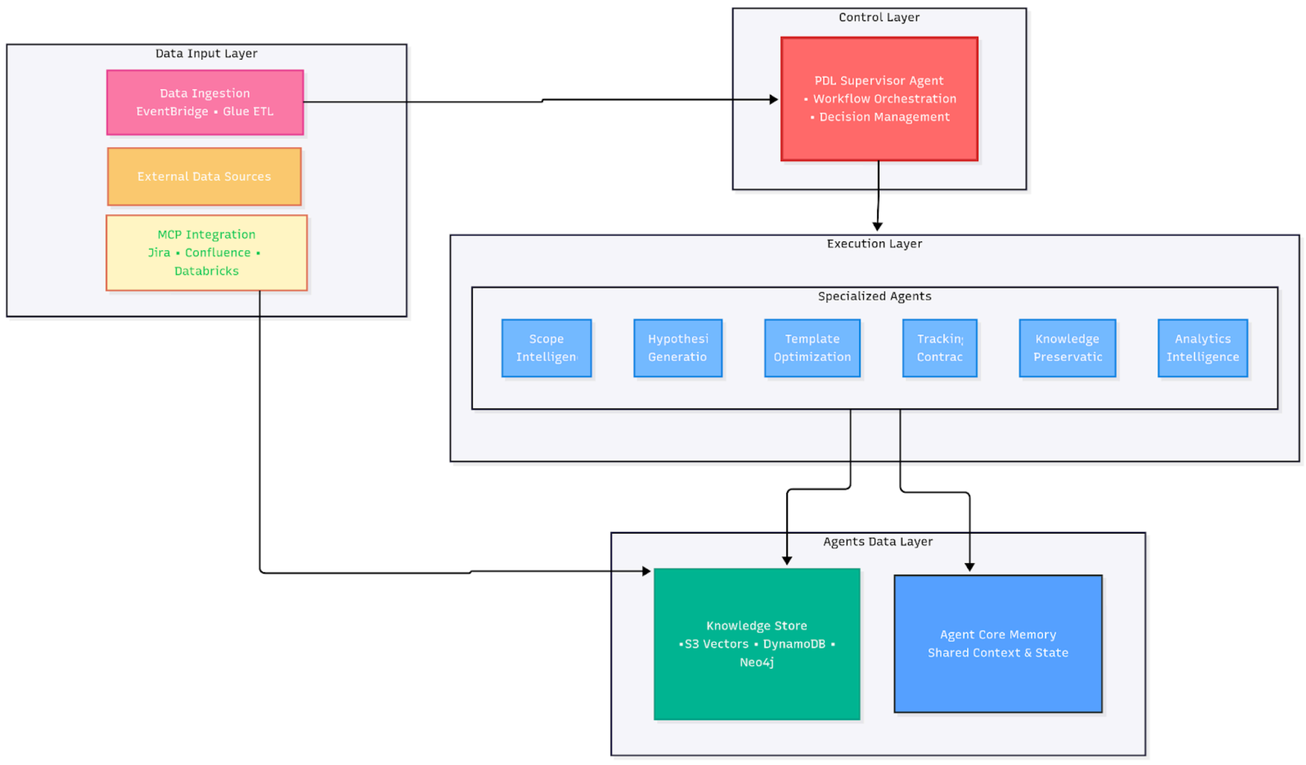
Games24x7 PDL Agents - Architecture

Top-Level Architecture



The PDL system follows a **4-layer horizontal architecture** designed for enterprise-scale product development lifecycle automation using AWS Agent Core and S3 vectors.

Data Input Layer

- **External Data Sources**: Aggregates data from Jira, Analytics platforms, GitHub, and CRM systems
- **MCP Integration**: Model Context Protocol connections to Databricks and Confluence for structured data access
- **Data Ingestion Pipeline**: EventBridge and Glue ETL for real-time and batch processing

Control Layer

- **PDL Supervisor Agent**: Master orchestrator managing entire workflow execution
- **Core Responsibilities**: Workflow orchestration, quality gate validation, and intelligent decision management
- **Agent Coordination**: Uses A2A protocol to coordinate specialized agents based on workflow requirements

Execution Layer

- **6 Specialized Agents**: Domain-specific agents handling scope intelligence, hypothesis generation, template optimization, tracking contracts, knowledge preservation, and analytics intelligence
- **Parallel Processing**: Agents can work simultaneously on different aspects of the same project
- **Quality Assurance**: Each agent validates outputs before passing to next workflow stage

Agents Data Layer

- **Knowledge Store**: S3 Vectors with DynamoDB and Neo4j for institutional knowledge, causal relationships, and metadata
- **Agent Core Memory**: Shared context and state management across all agents and workflows
- **Dual Access**: Both direct data ingestion and agent-processed information feed the knowledge repositories

Agent Specifications

1. PDL Supervisor Agent (Master Orchestrator)

Purpose: Central command and control for all PDL workflow orchestration

Core Responsibilities:

- **Workflow Orchestration**: Manage end-to-end PDL processes from scope creation to post-launch analysis
- **Agent Coordination**: Route tasks to specialized agents based on workflow stage and complexity
- **Decision Management**: Handle escalations and complex decisions requiring multi-agent input
- **Quality Assurance**: Validate outputs from specialized agents before proceeding to next workflow stage
- **Resource Optimization**: Balance workload across agents and manage priority queuing
- **Human Interface**: Provide single point of interaction for product managers and stakeholders

Orchestration Patterns:

- **Sequential Workflow**: Scope → Hypothesis → Template → Tracking → Analytics
- **Parallel Processing**: Multiple agents working on different aspects simultaneously
- **Adaptive Routing**: Dynamic agent selection based on workload and expertise
- **Rollback Management**: Ability to revert to previous workflow states

2. Scope Intelligence Agent

Purpose: Transform minimal PM inputs into comprehensive scope documents

Key Capabilities:

Input: Brief 2-3 paragraph scope document

Process:

```
├── Extract key concepts using Amazon Titan embeddings
├── Search S3 vectors for similar historical projects
├── Identify stakeholders based on project patterns
└── Generate contextualized scope document
```

Output: Standardized, comprehensive scope document

Data Dependencies:

- Historical project documents (S3 vectors)
- Team expertise mappings (DynamoDB)
- Template repository (S3 vectors)

3. Hypothesis Generation Agent

Purpose: Automate hypothesis creation using historical data and causal analysis

Core Technologies:

- **LLMCG (Large Language Model Causal Graphs)**: Extract causal relationships
- **Neo4j Graph Database**: Store 197K concepts, 235K causal connections
- **Jaccard Similarity**: Calculate hypothesis probability scores

Processing Flow:

Scope Document → Causal Graph Analysis → Historical Pattern Matching → KPI Alignment → Generated Hypotheses

-

4. Template Optimization Agent

Purpose: Eliminate template recreation through intelligent template management

Optimization Process:

1. **Document Clustering**: Group similar projects using vector similarity
2. **Pattern Extraction**: Identify successful template structures

3. **Template Synthesis**: Generate optimized templates from patterns
4. **Validation Loop**: Continuous improvement based on usage analytics

Repository Management:

- Version control with change tracking
- Usage analytics and success rate monitoring
- Auto-update mechanisms from new project patterns

5. Tracking Contract Agent

Purpose: Ensure complete tracking implementation through automated verification

Verification Methods:

- **Smart Contract Verification**: Mathematical proof of correctness
- **Runtime Verification**: Real-time monitoring with alerts
- **Gap Analysis**: Systematic comparison of required vs. implemented tracking
- **Compliance Checking**: Continuous monitoring against business rules

Early Warning System:

- Proactive alerting for missing tracking points
- Automated documentation generation
- Integration with analytics platforms for validation

6. Knowledge Preservation Agent

Purpose: Systematically capture and digitize tribal knowledge

Capture Methodologies:

- **Structured Documentation**: AI-powered conversion of informal knowledge
- **Interactive Extraction**: Conversational AI for expert interviews
- **Process Recording**: Automated workflow capture with transcription
- **Context-Aware Suggestions**: Real-time documentation prompts

Cultural Integration:

- Communities of practice facilitation
- Mentorship program coordination
- Knowledge sharing incentive tracking

7. Analytics Intelligence Agent

Purpose: Post-launch analysis and insight generation

Analysis Capabilities:

- **Performance Tracking**: Compare actual vs. predicted results

- **Pattern Recognition**: Identify success/failure patterns
- **Insight Generation**: Natural language explanations of findings
- **Recommendation Engine**: Suggest optimizations for future projects

Error Handling & Resilience

Agent Failure Recovery:

- Automatic failover to backup agent instances
- State recovery from persistent memory
- Graceful degradation with reduced functionality

Data Consistency:

- Event sourcing for all agent decisions
- Compensation patterns for failed operations
- Eventually consistent data propagation