High-Level Design (HLD)  
Project Delivery & Maintenance Platform

Version 1.0 • Generated on 21 Aug 2025

## Table of Contents (update in Word: References → Update Table)

1. Purpose & Scope

2. Architecture Goals & Quality Attributes

3. System Context

4. Use Cases (Overview)

5. Logical Architecture

6. Data & Integration Architecture

7. Deployment Architecture

8. Security Architecture

9. DevOps, CI/CD & Observability

10. Availability, Capacity & DR

11. Compliance & Audit

12. Risks & Decisions

13. Appendices (API, Data Model, Glossary)

# 1. Purpose & Scope

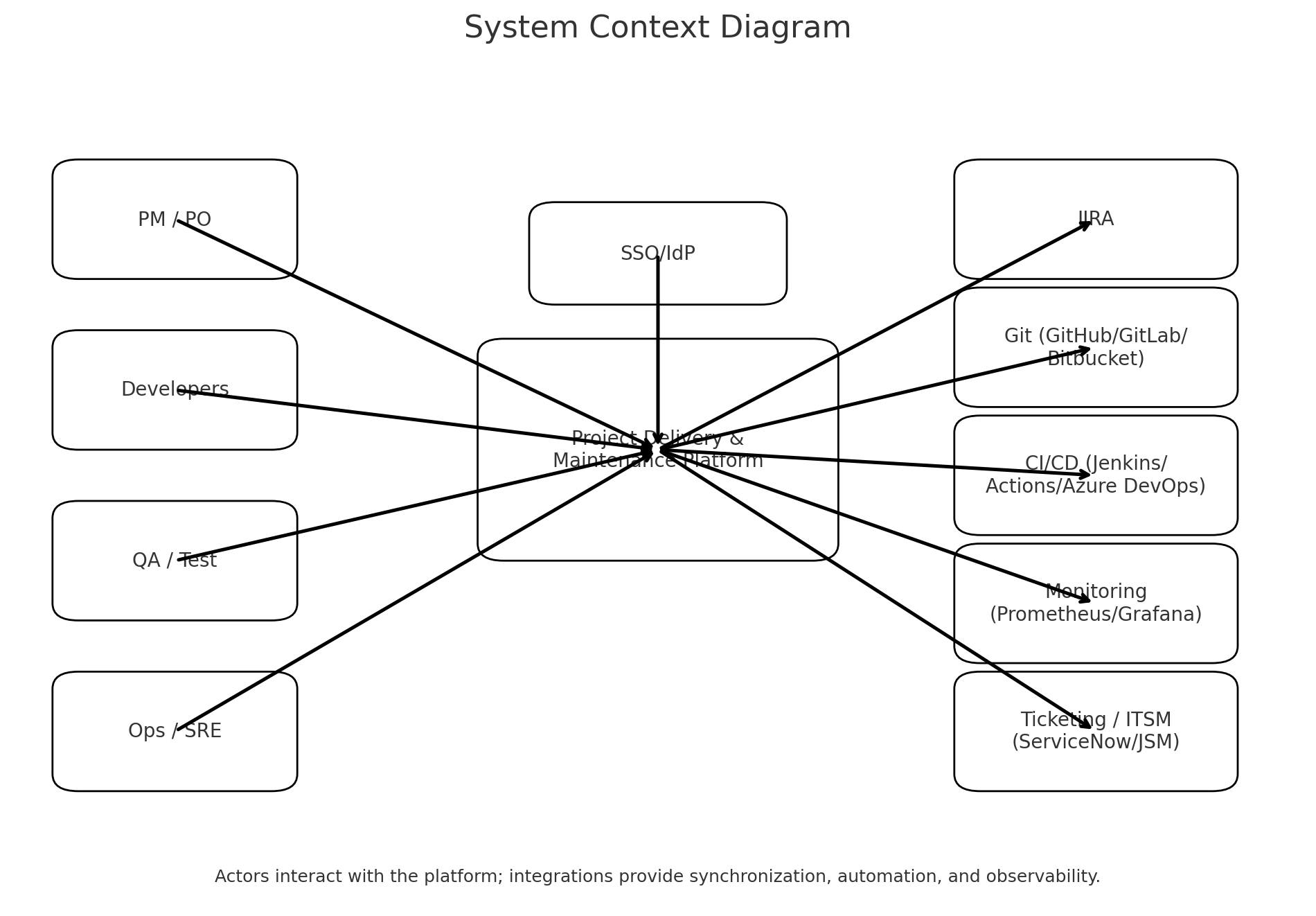
This HLD describes the target architecture of the Project Delivery & Maintenance Platform. It covers system context, core components, integrations, data, deployment topology, security, and operational concerns. The design is cloud-agnostic and integrates Agile and DevOps workflows.

# 2. Architecture Goals & Quality Attributes

* Cloud-agnostic deployment (AWS/Azure/GCP/on‑prem).
* Seamless JIRA/Git/CI/CD/ITSM integrations.
* Security by design: SSO (SAML/OIDC), RBAC, audit, encryption.
* Scalability to 1000+ concurrent users; <2s dashboard latency.
* Reliability: 99.9% uptime; blue/green or canary releases.
* Observability: logs, metrics, traces, SLOs and alerting.

# 3. System Context

Actors and external systems around the platform:

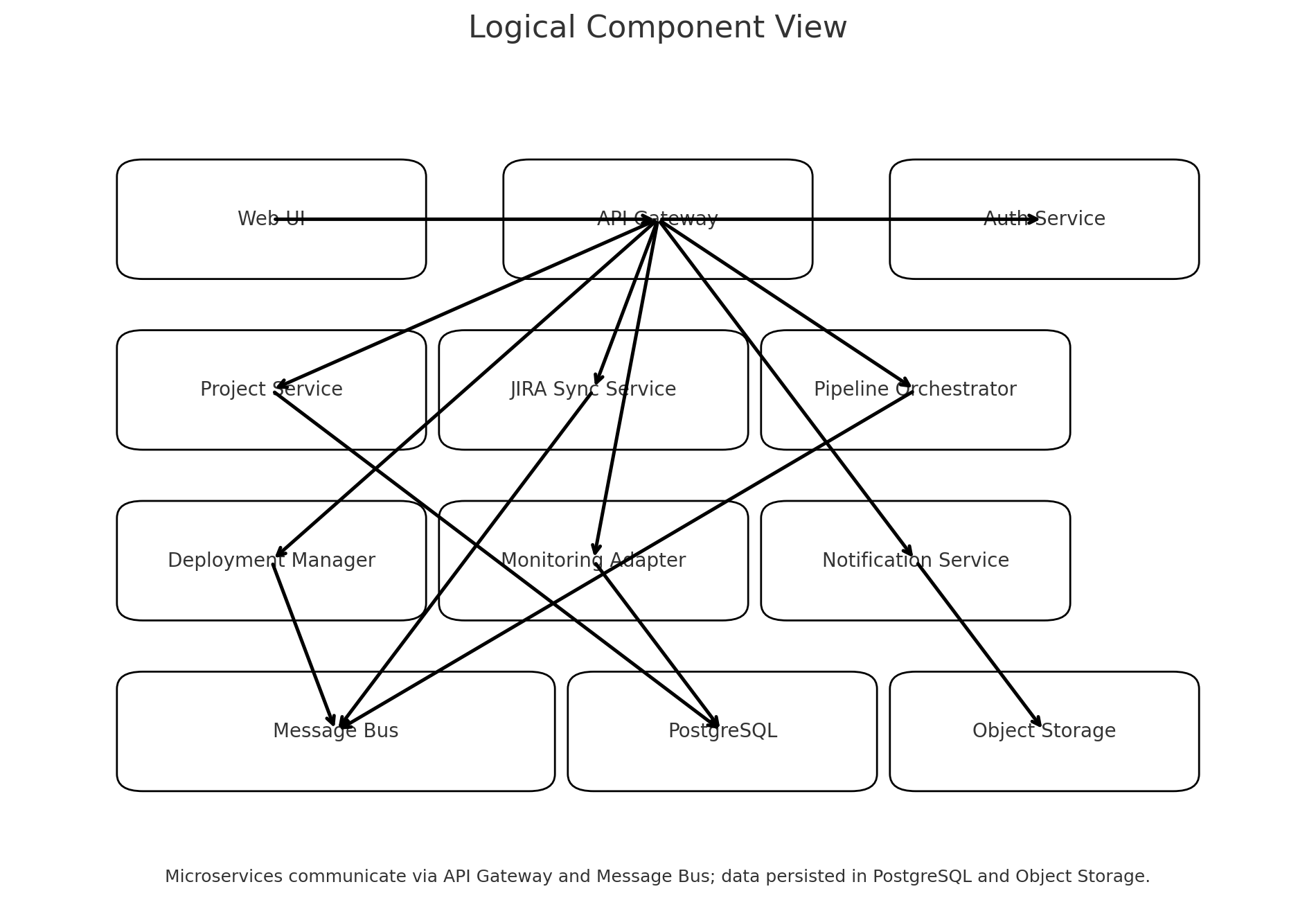


# 4. Use Cases (Overview)

* UC1: Manage backlog, sprints, and Kanban workflows.
* UC2: Sync epics/stories/tasks/bugs with JIRA bi‑directionally.
* UC3: Trigger CI/CD pipelines on commit/merge and track pipeline status.
* UC4: Deploy application workloads to multiple environments.
* UC5: Monitor health, receive alerts, and log incidents in ITSM.
* UC6: Role-based dashboards and reporting (velocity, lead time, MTTR).

# 5. Logical Architecture

The platform comprises a web UI, an API gateway, and a set of microservices: Project Service, JIRA Sync Service, Pipeline Orchestrator, Deployment Manager, Monitoring Adapter, Notification Service, and Auth Service. Services communicate via the gateway and a message bus; data is persisted in PostgreSQL and object storage.



# 6. Data & Integration Architecture

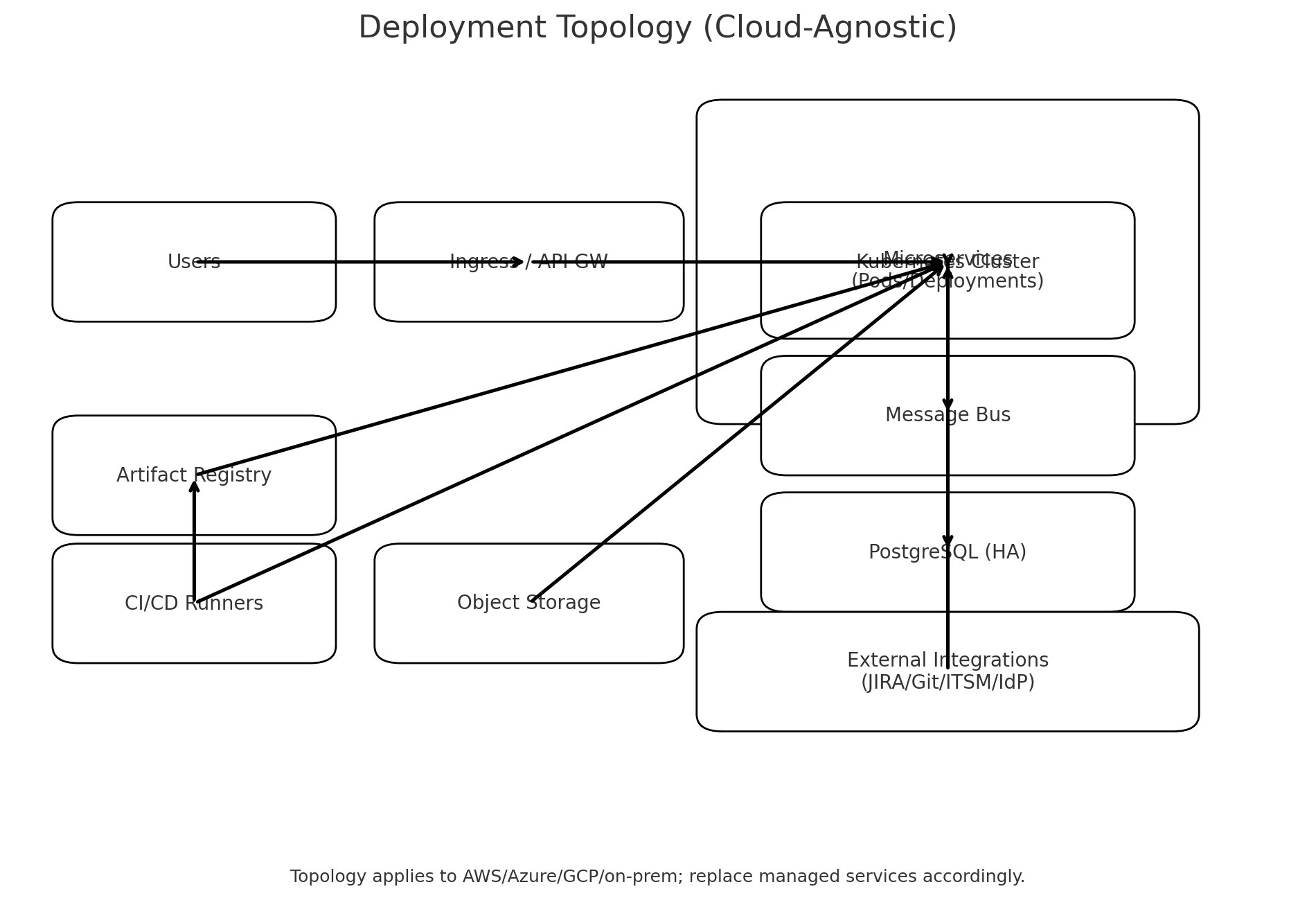
Core entities (high-level): Project, Epic, Story, Task, PipelineRun, Deployment, Incident, User, Role, Permission, IntegrationConfig.

Integration patterns: REST/GraphQL APIs, Webhooks, OAuth2/SAML for auth, scheduled sync jobs, idempotent upserts, and backoff/retry on failures.

Data retention & lineage: audit logs for changes; pipeline/deployment history retained per compliance policy.

# 7. Deployment Architecture

Kubernetes-based deployment with ingress and API gateway. Use Terraform and Helm for infrastructure and application provisioning. Replace managed services per provider: e.g., RDS/Cloud SQL/SQL MI for PostgreSQL; ECR/Artifact Registry/ACR for artifacts.

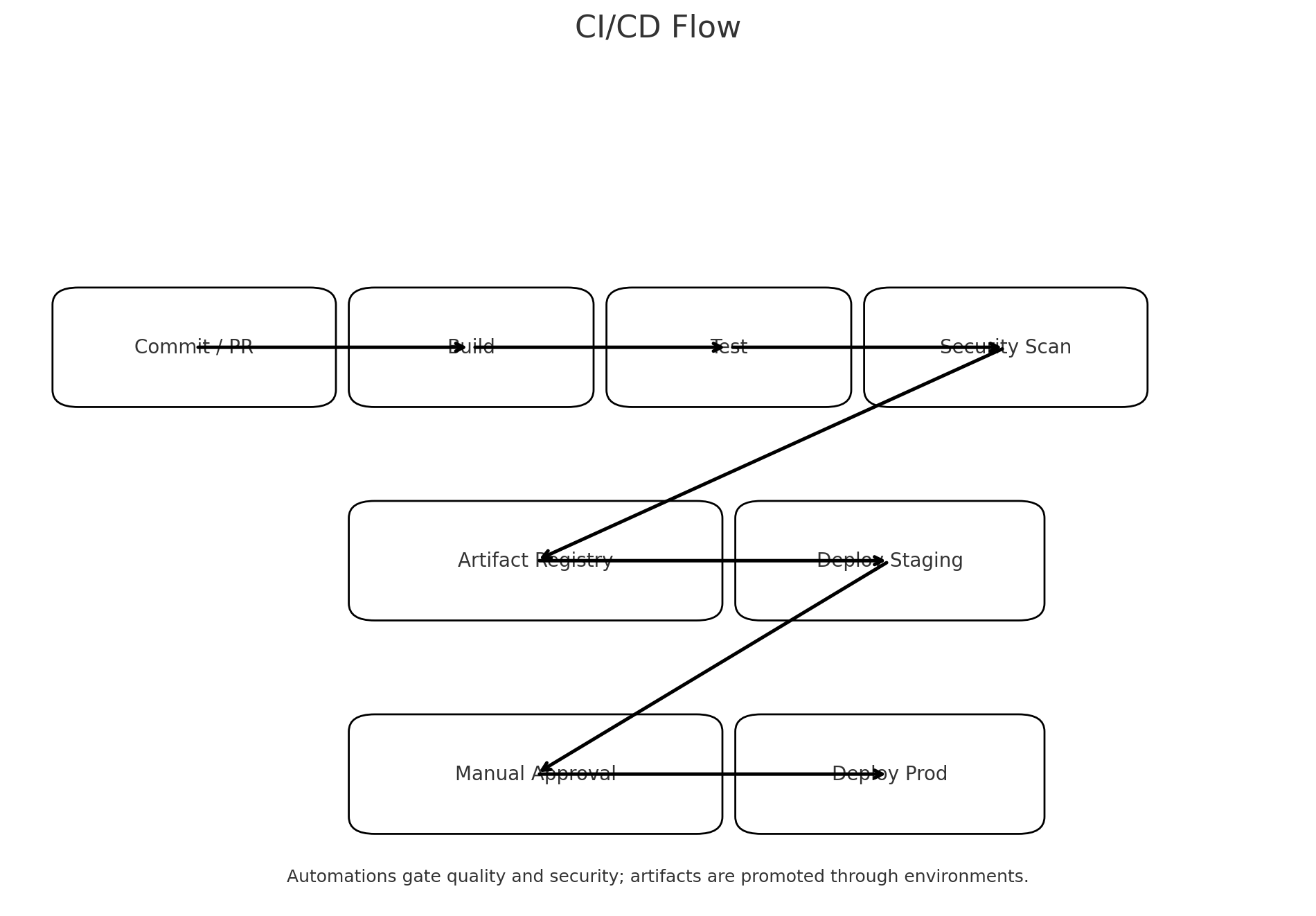


# 8. Security Architecture

* SSO via SAML/OIDC (Azure AD/Okta/Google).
* RBAC: roles for PM/PO/Dev/QA/Ops; least privilege and SoD.
* Secrets management via cloud KMS or HashiCorp Vault; no secrets in code.
* Encryption: TLS 1.2+ in transit; AES‑256 at rest for DB and object storage.
* Audit logging for admin actions, data exports, and configuration changes.
* Network controls: private cluster networking, WAF, and rate limiting at ingress.

# 9. DevOps, CI/CD & Observability

Pipeline stages and promotion strategy:



* Metrics: RED/USE metrics, deployment frequency, lead time, MTTR; surfaced in Grafana.
* Logging: structured logs (JSON) shipped to ELK/Cloud Logging; correlation IDs for traces.
* Tracing: OpenTelemetry to Jaeger/Tempo; trace IDs propagated via gateway.
* SLOs & Alerting: Error rate, latency, saturation thresholds with PagerDuty/MS Teams.

# 10. Availability, Capacity & DR

* HA via multi‑AZ clusters; horizontal pod autoscaling with resource quotas.
* Blue/green or canary deployments with automated rollback on health check failures.
* Backups: daily DB snapshots; object storage versioning; restore runbooks tested quarterly.
* Capacity planning: start with 3‑node control plane and 6‑12 worker nodes; scale by workload.

# 11. Compliance & Audit

* GDPR: data minimization; DSAR/RTBF procedures documented.
* SOC 2: change management, access reviews, incident response evidence artifacts.
* ISO 27001: risk register, asset inventory, backup and recovery controls.

# 12. Risks & Decisions

Key Risks:

* Integration rate limits (JIRA/Git/CI) may throttle sync jobs.
* Vendor API changes could break connectors; mitigate with contract tests and pinned SDK versions.
* Multi-cloud complexity; mitigate with IaC abstraction and golden templates.

Architectural Decisions (ADRs):

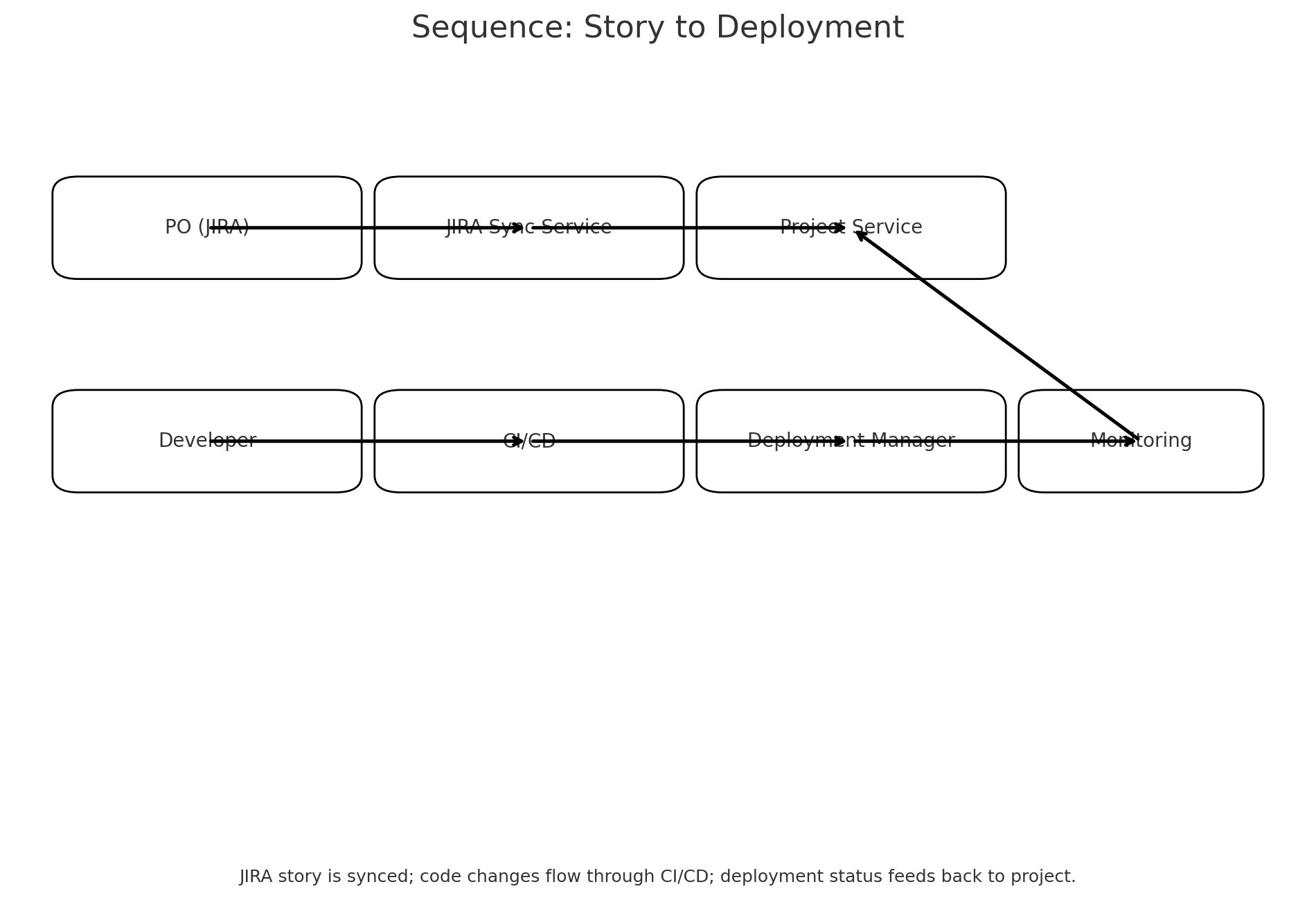
* ADR-001: Choose Kubernetes as the runtime for portability and scaling.
* ADR-002: Adopt API Gateway pattern for routing, auth, and rate limiting.
* ADR-003: Use PostgreSQL for relational data; object storage for binaries and exports.
* ADR-004: Standardize OpenTelemetry for observability across services.

# 13. Appendices

13.1 Example APIs (indicative):

* GET /api/projects/:id – fetch project summary
* POST /api/stories – create story (linked JIRA key)
* GET /api/pipelines/:id – pipeline run details
* POST /api/deployments – trigger deployment to env

13.2 Sequence Overview:



13.3 Glossary: CI/CD, MTTR, SLO, ADR, RBAC, OIDC, ITSM.