IEEE 42010:2022 Architecture Description

Model Context Protocol (MCP) Server

Mark Sigler

February 24, 2026

Contents

Chapter 1

Architecture Description: MCP Server

Document Identifier: MCP-AD-001
Version: 1.1.0
Date: 2025-07-17
Standard: ISO/IEC/IEEE 42010:2022
Status: Approved
Author: Mark Sigler

1.1 Document Control

Version	Date	Author	Changes
1.1.0	2025-07-17	Mark Sigler	Aligned with SRS v1.1.0: Streamable HTTP session management, expanded capability map, data model updates (outputSchema, annotations, icons), MCP protocol security best practices (NFR-SEC-073–081), updated ADR-005 to Streamable HTTP
1.0.0	2026-02-23	Mark Sigler	Initial AD derived from MCP-ARCHITECTURE v1.4.0, structured per IEEE 42010:2022

1.2 Table of Contents

1.3 1. AD Identification

1.3.1 1.1 Purpose

This Architecture Description (AD) describes the architecture of a production-ready Model Context Protocol (MCP) server. It is structured per ISO/IEC/IEEE 42010:2022 and realizes the requirements defined in the Software Requirements Specification (SRS).

1.3.2 1.2 Scope

The AD covers the architecture of a single MCP server instance and its supporting infrastructure:

- Five-layer enterprise architecture (Gateway, Server, Security, Observability, Integration)
- Core MCP primitives: resources, tools, prompts
- Advanced capabilities: sampling, elicitation, tasks
- Security framework: OAuth 2.1 + JWT/JWKS
- Containerized deployment on Docker and Kubernetes
- Observability pipeline: logging, metrics, tracing
- Multi-server orchestration patterns
- AI Service Provider Gateway

1.3.3 1.3 System of Interest

System: MCP Server
Version: Targets MCP specification 2025-11-25
Framework: FastMCP (Python)
Deployment: Containerized (Docker, Kubernetes)

1.3.4 1.4 References

Normative:

Reference	Location
SRS (IEEE 29148)	../IEEE-29148/SRS.md
MCP Specification 2025-11-25	https://modelcontextprotocol.io/docs/
MCP Authorization	https://modelcontextprotocol.io/docs/tutorials/security/authorization
JSON-RPC 2.0	https://www.jsonrpc.org/specification

Reference	Location
OAuth 2.1	https://oauth.net/2.1/
RFC 9728	https://datatracker.ietf.org/doc/html/rfc9728
FastMCP v3.x	https://gofastmcp.com/

Supporting Architecture Documents:

Document	Viewpoints Supported
01-architecture-overview.md	Functional
01b-architecture-decisions.md	All (ADRs)
02-security-architecture.md	Security
02a-data-privacy-compliance.md	Security, Information
02b-requirements-engineering.md	Development
03-tool-implementation.md	Functional
03a-prompt-implementation.md	Functional
03b-resource-implementation.md	Functional
03c-sampling-patterns.md	Functional
03d-decision-trees.md	Development
03e-integration-patterns.md	Functional
03f-elicitation-patterns.md	Functional
03g-task-patterns.md	Functional
03h-multi-server-orchestration.md	Functional, Deployment
03i-ai-service-provider-gateway.md	Functional, Deployment
04-testing-strategy.md	Development
05-observability.md	Operational
06-development-lifecycle.md	Development
06a-performance-scalability.md	Operational
07-deployment-patterns.md	Deployment
08-operational-runbooks.md	Operational
09-agentic-best-practices.md	Functional
10-migration-guides.md	Development
11-troubleshooting.md	Operational
12-cost-optimization.md	Operational
13-metrics-kpis.md	Operational
14-performance-benchmarks.md	Operational
15-mcp-protocol-compatibility.md	Development

1.4 2. Stakeholders and Concerns

1.4.1 2.1 Stakeholder Catalogue

Per IEEE 42010 §5.3, the following stakeholders have architecture concerns:

Stakeholder	Description	Key Concerns
Enterprise Architect	Technology governance authority	Standards compliance, integration patterns, maintainability
Engineering Lead	Technical decision authority	Feasibility, consistency, team velocity, code quality
Security Lead	Risk and compliance authority	Threat mitigation, access control, audit, data protection
DevOps Lead	Operations authority	Deployability, reliability, scalability, monitoring
Product Owner	Business priority authority	Feature availability, time-to-market, cost
Enterprise Developers	Build AI-powered applications	Clear APIs, documentation, debugging tools, testing
Platform Engineers	Manage deployments	Automation, container management, resource efficiency
Data Scientists	Create AI workflows	Data access, model integration, performance
Security Engineers	Ensure compliance	Vulnerability management, penetration testing, audit trails

1.4.2 2.2 Concern Catalogue

ID	Concern	Framing Stakeholders	Addressed by Viewpoints
CON-01	How are MCP primitives (tools, resources, prompts) structured?	Developers, Architect	Functional
CON-02	How does data flow through the system?	Developers, Data Scientists	Functional, Information
CON-03	How is data modeled and validated?	Developers, Data Scientists	Information

ID	Concern	Framing Stakeholders	Addressed by Viewpoints
CON-04	How is the system deployed and distributed?	DevOps, Platform Engineers	Deployment
CON-05	How are authentication and authorization enforced?	Security Lead, Security Engineers	Security
CON-06	How is the system monitored and debugged?	DevOps, Platform Engineers	Operational
CON-07	How is the codebase organized and tested?	Engineering Lead, Developers	Development
CON-08	How does the system scale under load?	DevOps, Architect	Operational, Deployment
CON-09	How are threats mitigated?	Security Lead	Security
CON-10	How is the system maintained and evolved?	Engineering Lead, Product Owner	Development
CON-11	How are AI providers integrated?	Developers, Architect	Functional, Deployment
CON-12	How are multiple MCP servers orchestrated?	Architect, Platform Engineers	Functional, Deployment

---

1.5 3. Viewpoints

Per IEEE 42010 §5.4, viewpoints are reusable conventions for constructing architectural views. This AD defines six viewpoints.

1.5.1 3.1 Functional Viewpoint

Purpose: Describe the runtime behavior and component structure of the MCP server.

Stakeholders: Developers, Architects, Data Scientists

Concerns addressed: CON-01, CON-02, CON-11, CON-12

Model kinds:

- Component diagram (C4 Level 2)
- Sequence diagram (request flow)
- Capability map

Analysis techniques:

- Scenario walkthrough
- Protocol compliance verification

1.5.2 3.2 Information Viewpoint

Purpose: Describe data structures, schemas, and information flow.

Stakeholders: Developers, Data Scientists

Concerns addressed: CON-03

Model kinds:

- Data model diagram
- Schema definitions (JSON Schema 2020-12)

Analysis techniques:

- Schema validation
- Data flow analysis

1.5.3 3.3 Deployment Viewpoint

Purpose: Describe how the system is packaged, distributed, and deployed.

Stakeholders: DevOps, Platform Engineers

Concerns addressed: CON-04, CON-08, CON-12

Model kinds:

- Container topology diagram
- Distribution flow diagram
- Infrastructure diagram

Analysis techniques:

- Capacity analysis
- Failure mode analysis

1.5.4 3.4 Security Viewpoint

Purpose: Describe trust boundaries, authentication, authorization, and threat mitigations.

Stakeholders: Security Lead, Security Engineers, Architect

Concerns addressed: CON-05, CON-09

Model kinds:

- Trust boundary diagram
- Authentication flow sequence
- RBAC model
- STRIDE threat model

Analysis techniques:

- Threat modeling (STRIDE)
- Attack surface analysis

1.5.5 3.5 Operational Viewpoint

Purpose: Describe monitoring, alerting, health checking, and performance management.

Stakeholders: DevOps, Platform Engineers

Concerns addressed: CON-06, CON-08

Model kinds:

- Observability pipeline diagram
- Health check topology
- Alert routing diagram

Analysis techniques:

- SLO/SLI analysis
- Capacity planning

1.5.6 3.6 Development Viewpoint

Purpose: Describe module structure, build pipeline, testing strategy, and development workflow.

Stakeholders: Engineering Lead, Developers

Concerns addressed: CON-07, CON-10

Model kinds:

- Module structure diagram
- CI/CD pipeline diagram
- Testing pyramid

Analysis techniques:

- Dependency analysis
- Coverage analysis

1.6 4. Views

1.6.1 4.1 Functional View

1.6.1.1 4.1.1 Five-Layer Architecture

The MCP server operates within a five-layer enterprise architecture. Each layer has distinct responsibilities with clear interfaces.

Layer responsibilities (see architecture overview for full detail):

Layer	Responsibility	SRS Requirements
Gateway	TLS termination, auth validation, routing, session management (MCP-Session-Id, MCP-Protocol-Version), global rate limits, correlation IDs, circuit breaker coordination, DNS rebinding protection	FR-PROTO-005, FR-PROTO-006, FR-PROTO-006a, FR-PROTO-025–031, NFR-SEC-022, NFR-PERF-017
Server	**FastMCP v3.x Server**: Tool execution, resource serving, prompt rendering, sampling, elicitation, task management, list_changed notifications, lifecycle management; **Middleware Stack**: authorization, rate limiting, caching, logging, timing; **Provider System**: component sourcing from decorators, filesystem, proxies, OpenAPI specs; **Dependency Injection**: request context, tokens, custom deps	FR-TOOL-*, FR-RSRC-*, FR-PROMPT-*, FR-SAMP-*, FR-ELIC-*, FR-TASK-*, FR-PROTO-032–034, NFR-OBS-*, NFR-SEC-025–026
Security	OAuth 2.1 + PKCE, JWT verification, RBAC, capability ACL, input validation, audit logging, MCP protocol security (confused deputy, SSRF, session hijacking, scope minimization)	NFR-SEC-001–081
Observability	Structured JSON logging, Prometheus metrics, OpenTelemetry tracing, health checks	NFR-OBS-001–013
Integration	External API calls, database access, legacy system adapters	FR-TOOL-011–015

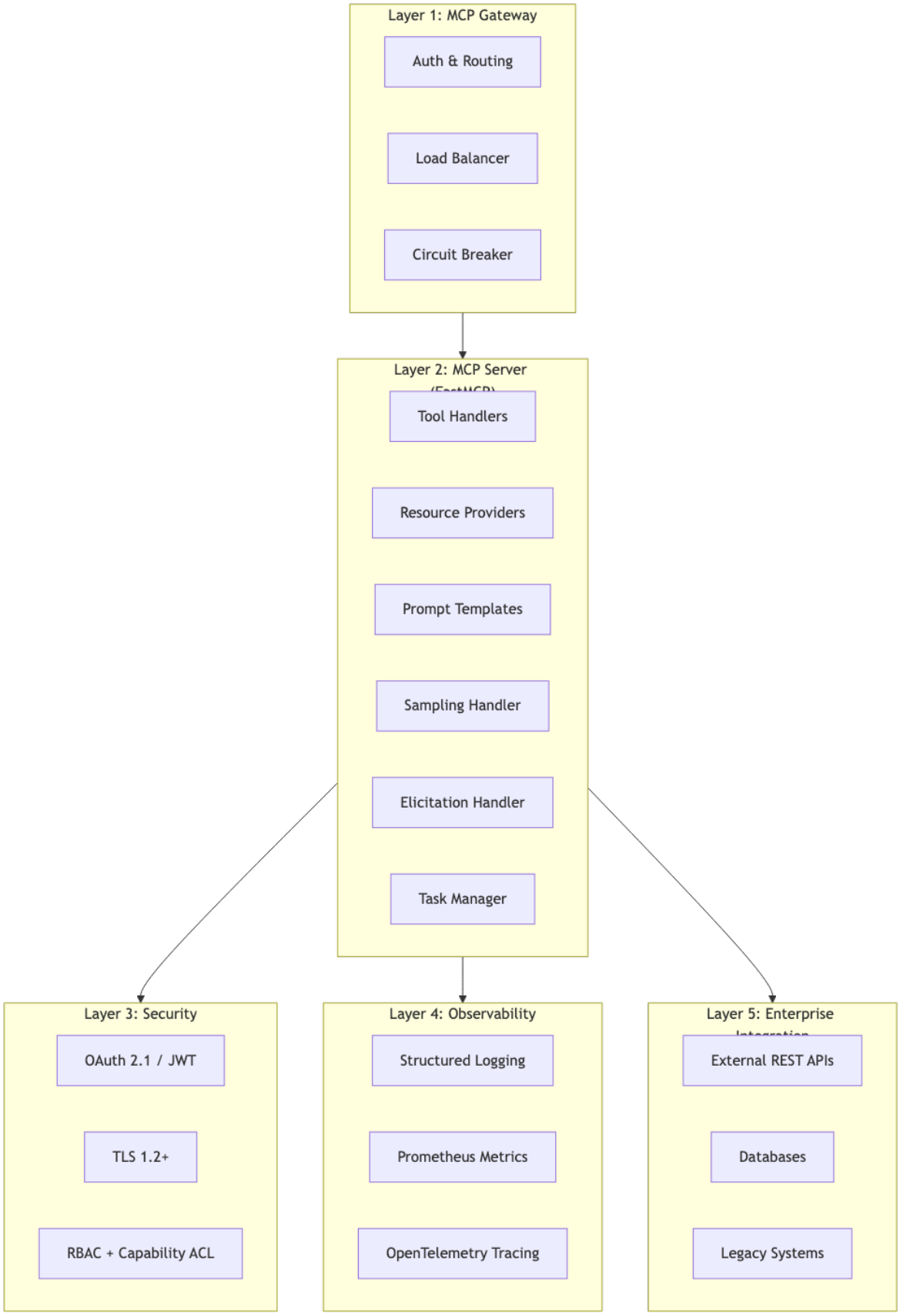

Figure 1.1: Diagram 1

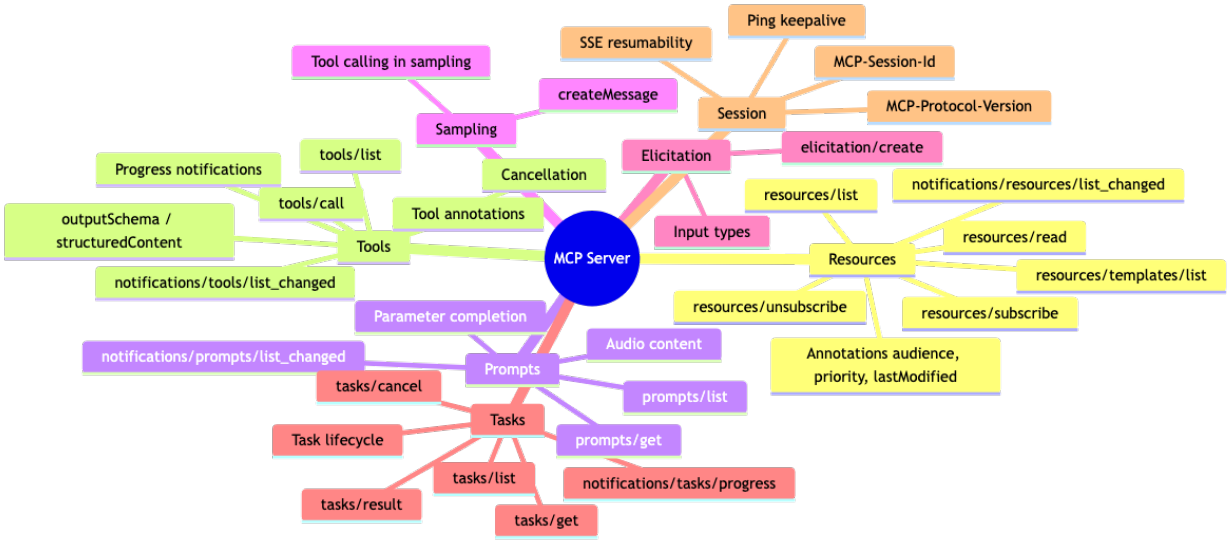

Figure 1.2: Diagram 2

1.6.1.2 4.1.2 MCP Capability Map

1.6.1.3 4.1.3 Request Flow

Timing budget:

Step	Budget
Gateway processing	10–20 ms
JWT validation (cached)	5–15 ms
Input validation	1–5 ms
Backend integration	100–300 ms (variable)
Protocol overhead	$<$ 50 ms
Total p95	**$<$ 500 ms**

1.6.1.4 4.1.4 Multi-Server Orchestration

Clients compose multiple MCP servers for cross-domain workflows. Each server declares clear capability boundaries (FR-ORCH-001). Servers never directly access other servers; cross-server coordination is client-orchestrated (FR-ORCH-005).

1.6.1.5 4.1.5 AI Service Provider Gateway

The gateway abstracts AI provider differences, enabling provider-agnostic deployments (CP-03).

Gateway capabilities:

- Configurable `base_url` for any OpenAI-compatible endpoint (FR-GWWY-001)
- Automated `/v1/models` connectivity handshake (FR-GWWY-002)
- Enterprise header injection (`X-Project-ID`, `X-Cost-Center`) (FR-GWWY-003)
- Automatic failover on 429/5xx/timeout (FR-GWWY-004–006)
- Model ID mapping and per-instance rate limits (FR-GWWY-008–009)
- Encrypted credential storage (FR-GWWY-007, NFR-SEC-058–063)

1.6.2 4.2 Information View

1.6.2.1 4.2.1 Core Data Model

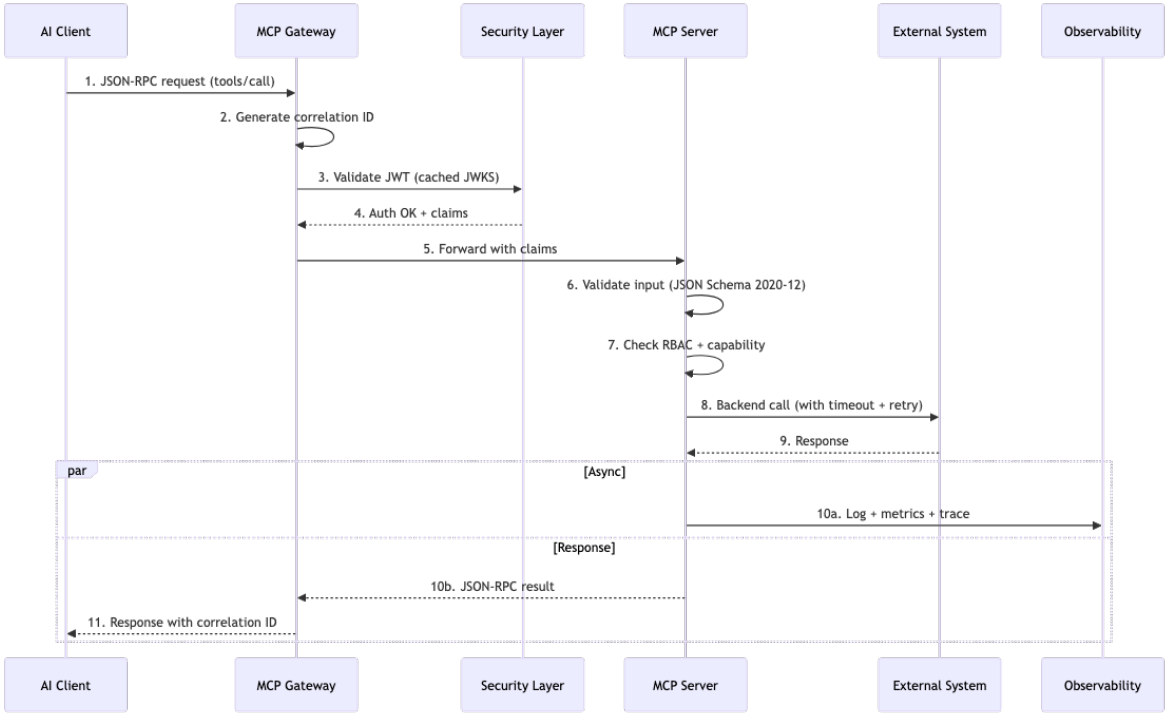

Figure 1.3: Diagram 3

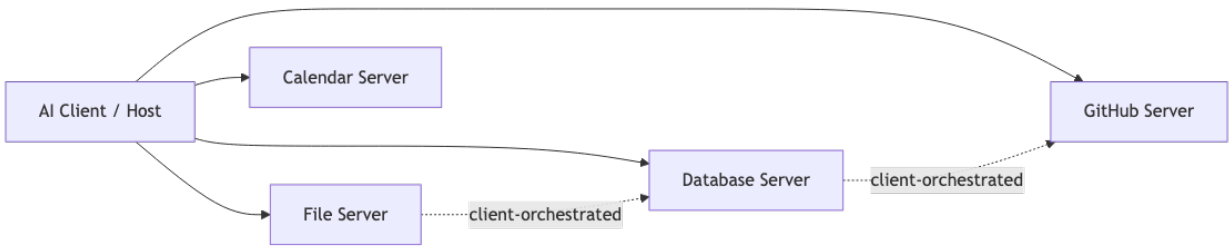

Figure 1.4: Diagram 4

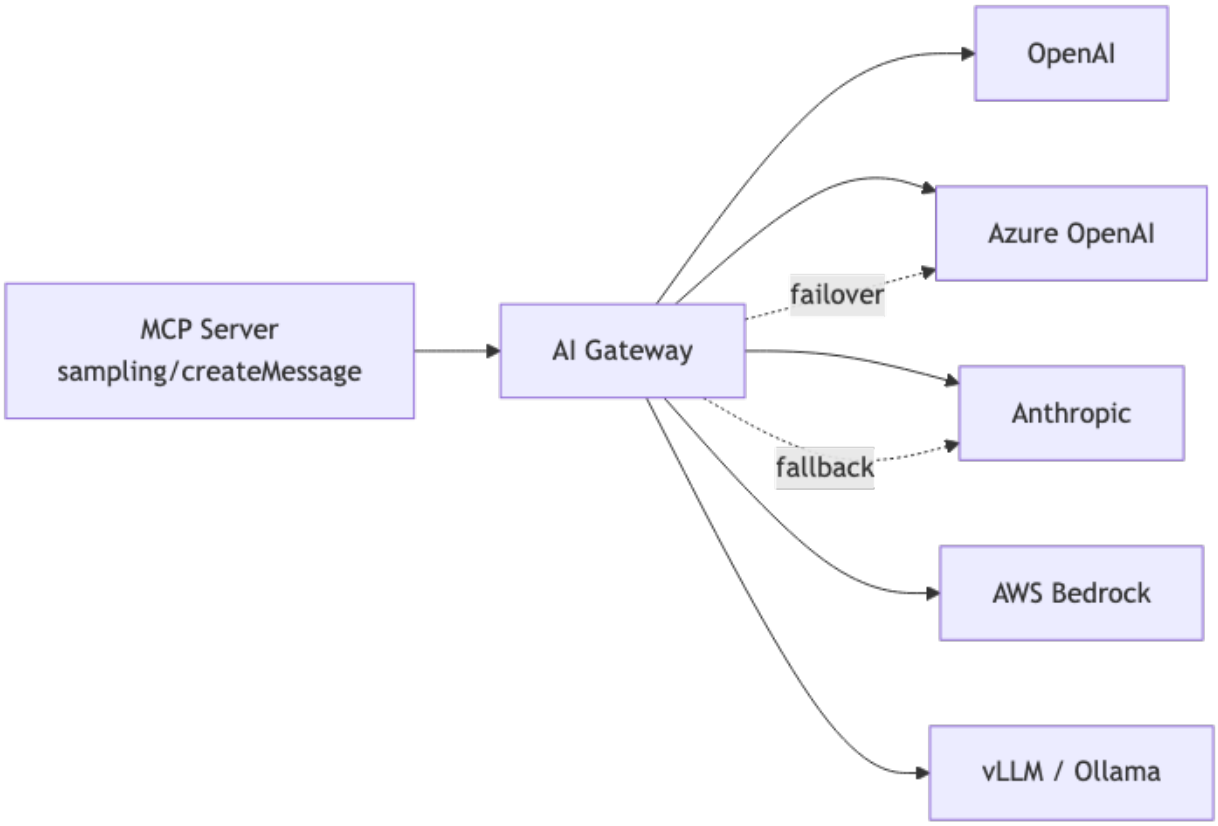

Figure 1.5: Diagram 5

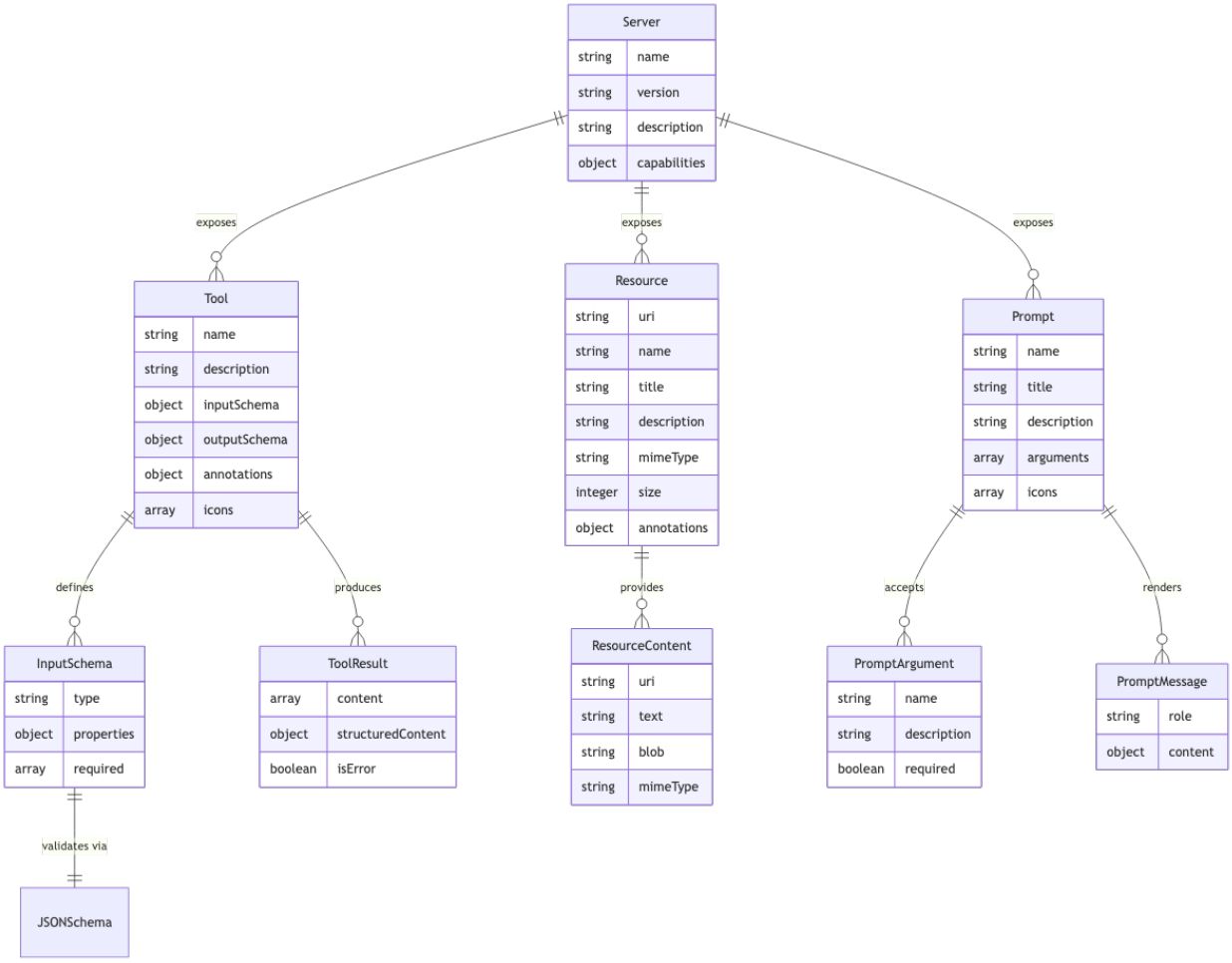

Figure 1.6: Diagram 6

JSON Schema 2020-12 is used for all tool input validation (DC-003). Schemas are declared in `tools/list` responses and enforced before tool execution (FR-TOOL-005).

1.6.2.3 4.2.3 Audit Event Structure

```
{
  "timestamp": "2025-11-20T10:30:00Z",
  "event_type": "tool_execution",
  "user_id": "user@example.com",
  "user_role": "developer",
  "action": "tools/call:get_forecast",
  "result": "success",
  "ip_address": "10.0.1.42",
  "correlation_id": "550e8400-e29b-41d4-a716-446655440000",
  "duration_ms": 245
}
```

1.6.3 4.3 Deployment View

1.6.3.1 4.3.1 Container Topology

Topology Legend:

- **Solid lines (→)**: Runtime dependencies (network calls, data flow)
- **Dashed lines (-.→)**: Configuration/control relationships (mounts, scaling)
- **Cylinders**: Data stores (databases, key-value stores)
- **Rectangles**: Compute resources (pods, services)

1.6.3.2 4.3.2 Container Specification

Property	Value	SRS Requirement
Base image	Alpine / distroless	NFR-CNTR-001
Compressed size	< 100 MB	NFR-CNTR-002
Architectures	AMD64, ARM64	NFR-CNTR-003
User	Non-root (UID > 1000)	NFR-CNTR-006
Filesystem	Read-only root	NFR-CNTR-007
Capabilities	All dropped, only required added	NFR-CNTR-008
Shell / pkg-mgr	None in final image	NFR-CNTR-009
Secrets in layers	None	NFR-CNTR-010
Vulnerability scan	Zero critical/high	NFR-CNTR-004
CIS Benchmark	Passes	NFR-CNTR-012
Startup time	< 5 seconds	NFR-CNTR-005

Base Image Decision (ADR-007):

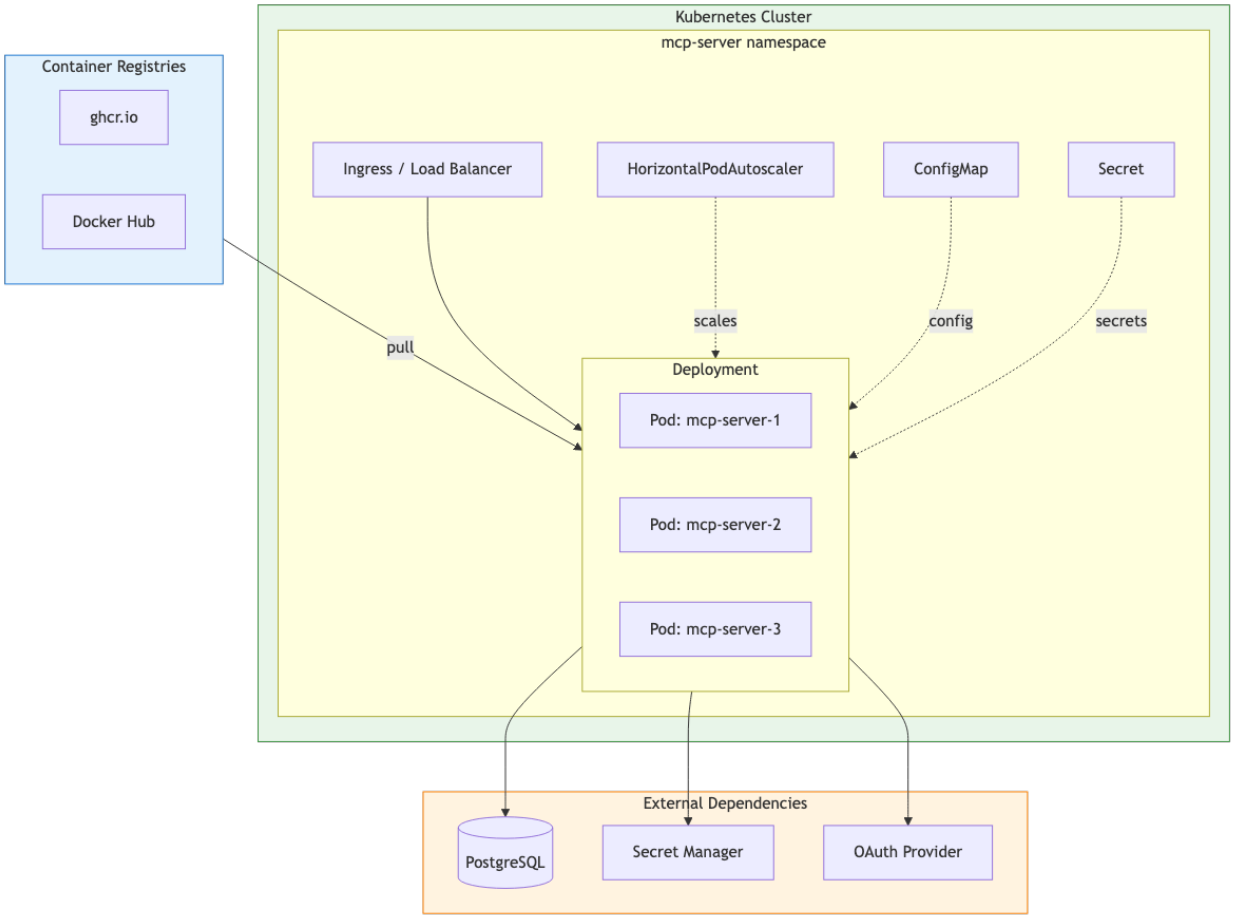

Figure 1.7: Diagram 7

- **Production (recommended)**: `gcr.io/distroless/python3-debian12` — No shell, no package manager, minimal attack surface (5-20MB), zero OS CVE exposure, strongest compliance posture (PCI-DSS, SOC2, NIST 800-190)
- **Development/Debug**: `python:3.11-alpine` — Shell access for troubleshooting, `apk` for build-time dependencies, 20-50MB size, musl libc (note: some Python packages require glibc workarounds)
- **Multi-stage builds**: Alpine for build stage (compile extensions, install deps) → distroless for runtime stage (copy artifacts, no tooling)

Trade-offs:

Aspect	Distroless	Alpine
Attack Surface	Minimal (no shell/pkg-mgr)	Small (busybox, apk)
Size	5-20 MB	20-50 MB
CVE Exposure	OS packages: 0	OS packages: ~5-15
Debugging	Limited (no shell)	Full (sh, common utils)
Build Complexity	Requires multi-stage	Single-stage possible
Python Compatibility	Standard glibc	musl libc (edge cases)
Compliance	Strongest	Strong
Image signing	Cosign or Docker Content Trust	NFR-CNTR-027
SBOM	Included	NFR-CNTR-028
Provenance	Attestation included	NFR-CNTR-029

1.6.3.3 4.3.3 Volume Mount Convention

Mount Path	Access	Purpose	SRS Requirement
`/config`	Read-only	Configuration files	NFR-CNTR-014, NFR-CNTR-017
`/data`	Read-write	Persistent application data	NFR-CNTR-017
`/secrets`	Read-only	Mounted secrets	NFR-CNTR-015, NFR-CNTR-017
`/logs`	Write	Log output	NFR-CNTR-017

1.6.3.4 4.3.4 Configuration Hierarchy

Per 12-factor app (DC-011):

1. **Environment variables** — primary source (NFR-CNTR-013)
2. **Mounted config files** — override defaults (NFR-CNTR-014)
3. **Secret manager references** — credentials (NFR-CNTR-015)

All configuration validated on startup with clear error messages (NFR-CNTR-016).

1.6.3.5 4.3.5 Distribution Flow

Artifact Types:

- **Container Image**: Built and scanned container (base artifact)

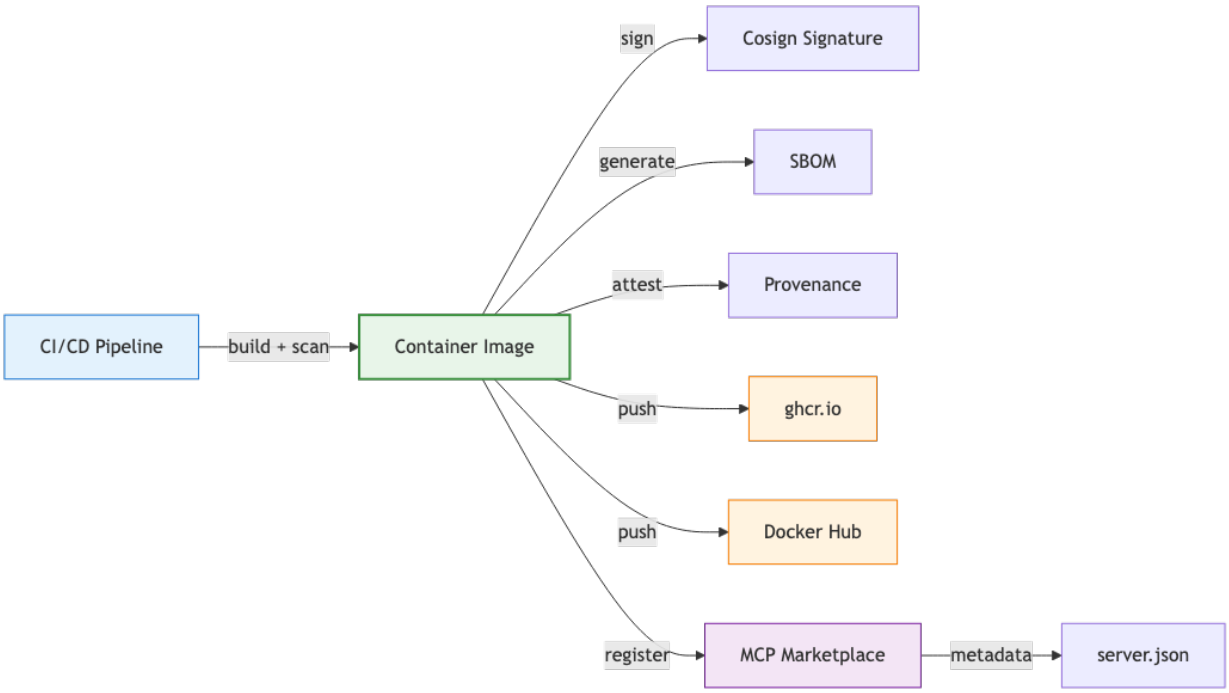

Figure 1.8: Diagram 8

- **Signature**: Cryptographic signature (Cosign/DCT) for verification
- **SBOM**: Software Bill of Materials (CycloneDX/SPDX format)
- **Provenance**: Build attestation (SLSA provenance)
- **Registry**: Public/private container registries (ghcr.io, Docker Hub)
- **Marketplace**: MCP server registry with server.json metadata

Publish within 10 minutes of release (NFR-CNTR-030).

1.6.3.6 4.3.6 Scaling

Dimension	Mechanism	SRS Requirement
Horizontal	Stateless replicas via HPA	NFR-PERF-010
Concurrency	100+ connections per instance	NFR-PERF-007
Throughput	$>$ 100 req/s	NFR-PERF-011
Memory	$<$ 500 MB baseline per instance	NFR-PERF-009
Graceful shutdown	SIGTERM within 30 s	NFR-CNTR-020

1.6.4 4.4 Security View

1.6.4.1 4.4.1 Trust Boundaries

Trust Zone Definitions:

- **Untrusted Zone (Red)**: Public internet, no security assumptions
- **DMZ (Orange)**: Perimeter defense, TLS termination, limited trust
- **Trusted Zone (Green)**: Authenticated services, application logic
- **Internal Services (Blue)**: Data stores, highest security, no external access
- **Auth Provider (Purple)**: Identity and access management, federated trust

Role	Capabilities	SRS Requirement
service	Limited programmatic access (configured per-service)	NFR-SEC-017

Authorization policy: deny-by-default (NFR-SEC-019). Every request validated against role + capability (NFR-SEC-021).

1.6.4.4 4.4.4 Defense in Depth Layers

Layer	Controls	SRS Requirements
Network	TLS 1.2+, Origin validation, CORS allowlist	NFR-SEC-043, FR-PROTO-006, NFR-SEC-057
Authentication	OAuth 2.1 + PKCE, JWT/JWKS, API keys	NFR-SEC-001–016
Authorization	RBAC (4 roles), capability ACL, deny-by-default	NFR-SEC-017–021
Application	Input validation (7 attack classes), rate limiting, security headers	NFR-SEC-030–057
Data	PII masking, encrypted secrets, TLS in transit	NFR-SEC-041–045, NFR-SEC-058–065
MCP Protocol	Confused deputy mitigation, token passthrough prohibition, SSRF protections (private IP blocking, HTTPS enforcement), session hijacking prevention, scope minimization, consent cookie hardening	NFR-SEC-073–081
Audit	JSON audit events, append-only, 1-year retention	NFR-SEC-046–050

1.6.4.5 4.4.5 Security Headers

Every HTTP response includes (NFR-SEC-051–057):

```
Content-Security-Policy: default-src 'self'
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Referrer-Policy: no-referrer
```

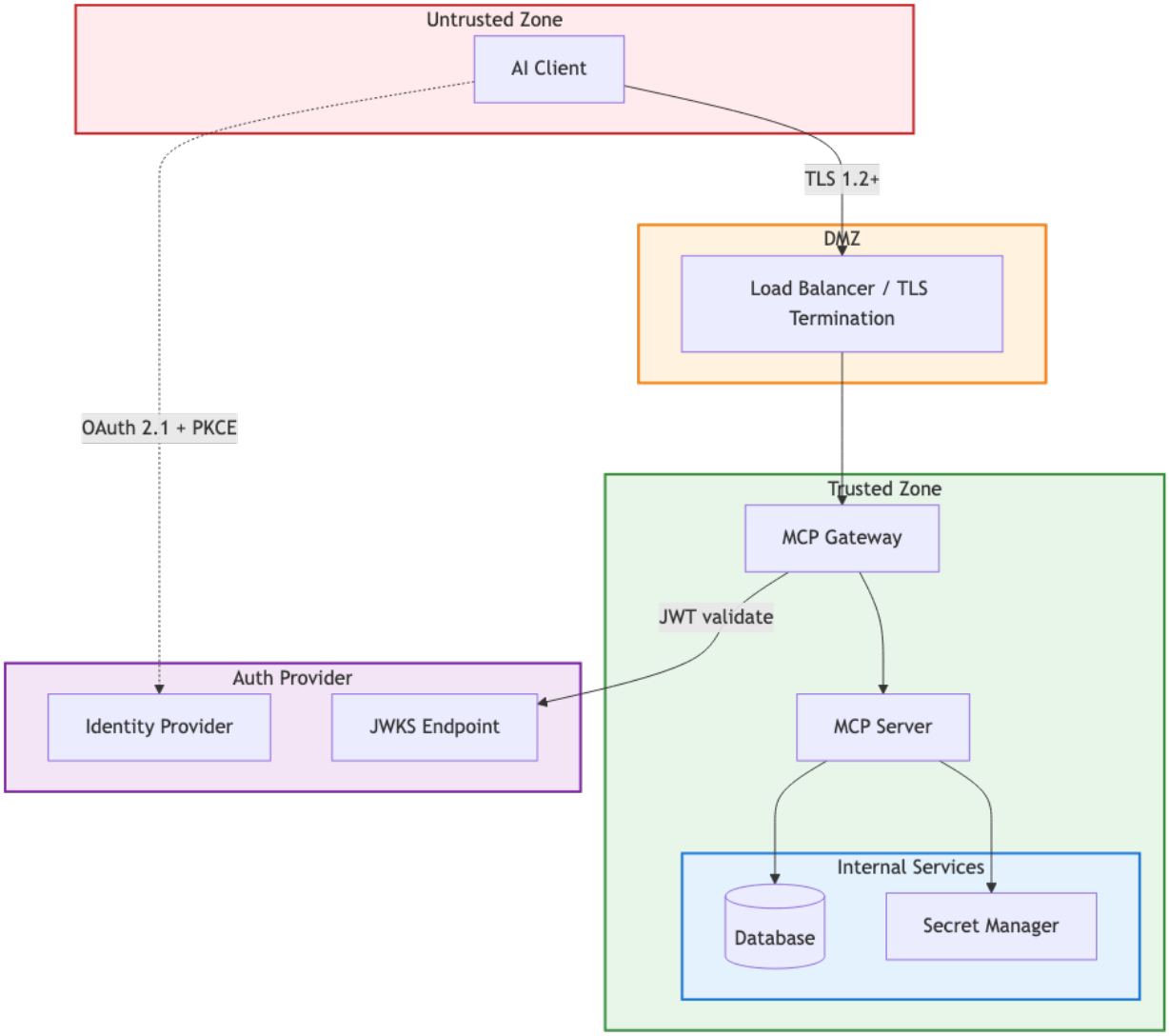

Figure 1.9: Diagram 9

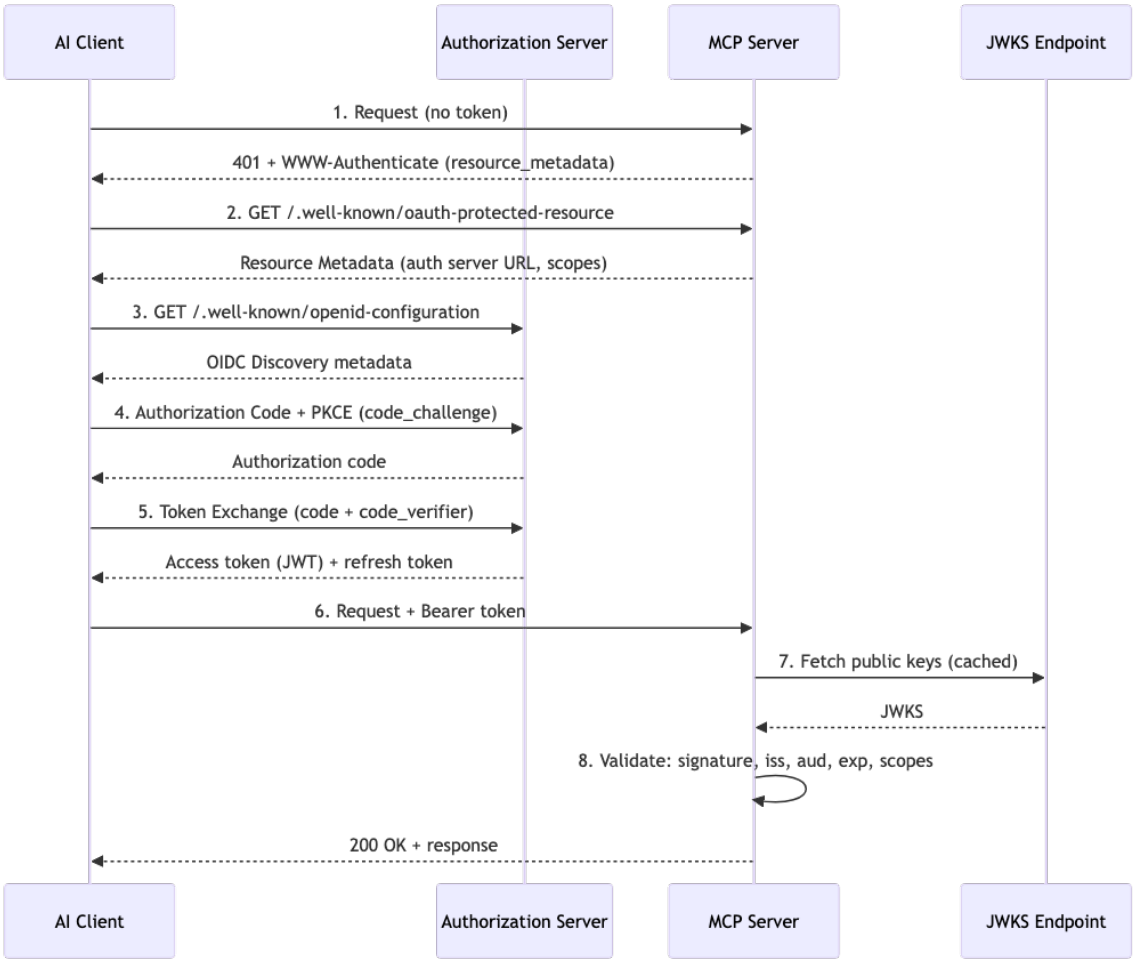

Figure 1.10: Diagram 10

1.6.4.6 4.4.6 Input Validation Matrix

Attack Class	Mitigation	SRS
SQL Injection	Parameterized queries only	NFR-SEC-030
Command Injection	Subprocess with argument lists, no shell	NFR-SEC-031
Path Traversal	Validate against base directory whitelist	NFR-SEC-032
XSS	Sanitize output, HTML-encode	NFR-SEC-033
XXE	Disable external entities	NFR-SEC-034
SSRF	URL whitelist validation	NFR-SEC-035
ReDoS	Regex complexity limits + timeouts	NFR-SEC-036

Request limits: 1 MB max size (NFR-SEC-037), 5 levels JSON depth (NFR-SEC-038), 10,000 char string max (NFR-SEC-039), 30 s timeout (NFR-SEC-040).

1.6.5 4.5 Operational View

1.6.5.1 4.5.1 Observability Pipeline

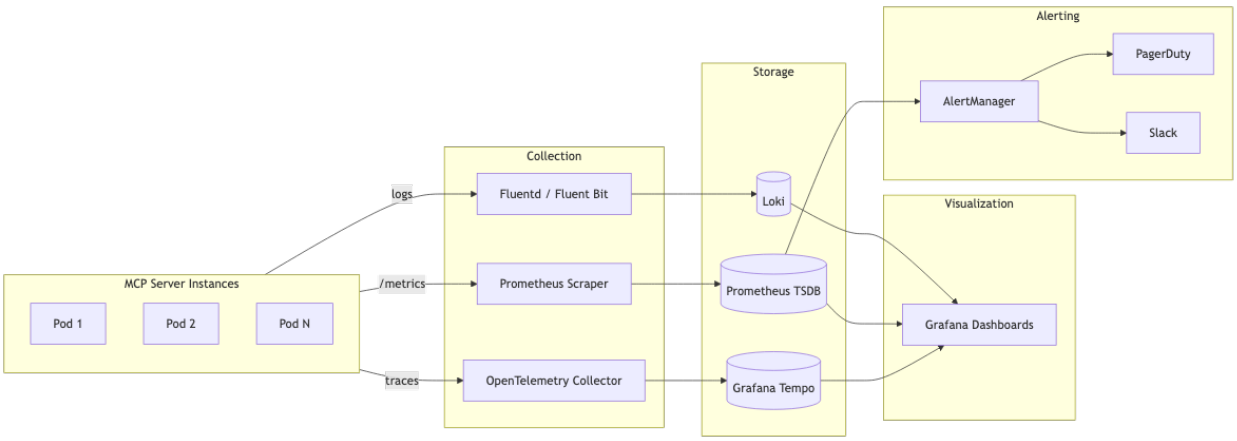

Figure 1.11: Diagram 11

1.6.5.2 4.5.2 Metrics Exposition

Prometheus-compatible endpoint at `/metrics` (NFR-OBS-005):

Metric	Type	Labels	SRS
`mcp_requests_total`	Counter	endpoint, method, status	NFR-OBS-006
`mcp_request_duration_seconds`	Histogram	endpoint, quantile	NFR-OBS-006
`mcp_errors_total`	Counter	endpoint, error_type	NFR-OBS-006

Metric	Type	Labels	SRS
`mcp_active_connections`	Gauge	—	NFR-OBS-006
`mcp_rate_limit_hits_total`	Counter	tier	NFR-OBS-006

1.6.5.3 4.5.3 Health Check Topology

Endpoint	Probe Type	Response Time	Checks	SRS
`/health`	Liveness	< 100 ms	Server running	NFR-OBS-008
`/ready`	Readiness	< 500 ms	Dependencies available	NFR-OBS-009
`/startup`	Startup	< 5 s	Initialization complete	NFR-OBS-010

1.6.5.4 4.5.4 Performance Budget

Metric	Target	SRS
p50 response time	< 100 ms	NFR-PERF-012
p95 response time	< 500 ms	NFR-PERF-002
p99 response time	< 1000 ms	NFR-PERF-013
Protocol overhead	< 50 ms	NFR-PERF-004
Auth latency (cached)	< 100 ms	NFR-PERF-005
Gateway overhead	< 30 ms	NFR-PERF-006
Error rate	$< 0.1\%$	NFR-PERF-014
Availability	99.9%	NFR-PERF-023

1.6.5.5 4.5.5 Resilience Patterns

Pattern	Configuration	SRS
Circuit breaker	Open after 5 failures, half-open after 30 s	NFR-PERF-017
Retry with backoff	1 s, 2 s, 4 s, 8 s exponential	NFR-PERF-020
Failover (429)	Immediate to secondary	NFR-PERF-019
Failover (5xx)	After backoff, within 500 ms total	NFR-PERF-020
Failover (timeout)	Immediate, within 50 ms	NFR-PERF-021
Failure detection	Within 200 ms	NFR-PERF-022
Graceful degradation	Continue serving available capabilities	NFR-PERF-015

1.6.5.6 4.5.6 Rate Limiting Tiers

Tier	Limit	SRS
Global	1000 req/min	NFR-SEC-022
Per-user	60 req/min	NFR-SEC-023

Tier	Limit	SRS
Per-API-key	120 req/min	NFR-SEC-024
Per-endpoint	Configurable	NFR-SEC-025

Algorithm: token bucket with burst support (NFR-SEC-026). All limits configurable per deployment (NFR-SEC-029).

1.6.6 4.6 Development View

1.6.6.1 4.6.1 Module Structure

```
mcp-server/
  src/
      server.py            # FastMCP server setup + initialization
      tools/               # Tool implementations (verb_noun.py)
      resources/           # Resource providers (uri_scheme.py)
      prompts/             # Prompt templates
      auth/                # OAuth 2.1 + JWT middleware
      middleware/           # Rate limiting, logging, headers
      models/              # Data models and schemas
      utils/               # Shared utilities
  tests/
      unit/                # Unit tests (> 80% coverage)
      integration/         # Integration tests (> 70%)
      contract/            # MCP protocol contract tests (100%)
      security/            # Security-specific tests
      performance/         # Load and latency tests
  config/                   # Configuration files
  docs/                     # Architecture & requirements documentation
  Dockerfile                # Multi-stage production build
  docker-compose.yml        # Local development compose
  helm/                     # Kubernetes Helm chart
  .github/workflows/        # CI/CD pipelines
```

1.6.6.2 4.6.2 CI/CD Pipeline

Figure 1.12: Diagram 12

Quality gates:

- Pre-commit: lint + format + unit tests pass
- PR: integration + contract + SAST + security tests pass, coverage thresholds met
- Main: full suite + container scan (zero critical/high), image built + signed

- Release: performance + E2E pass, publish to ghcr.io + Docker Hub + MCP Marketplace within 10 min

1.6.6.3 4.6.3 Testing Pyramid

Level	Scope	Coverage Target	SRS
Unit	Functions, classes	> 80% (tools: 90%)	§5.3 SRS
Integration	API endpoints, data flows	> 70%	§5.3 SRS
Contract	MCP protocol compliance	100%	§5.3 SRS
Security	Auth, validation, injection	100% critical paths	§5.3 SRS
Performance	Load, latency	Key endpoints	§5.3 SRS
E2E	User workflows	Critical paths	§5.3 SRS

1.6.6.4 4.6.4 Framework Choice

FastMCP v3.x selected per ADR-002:

- **Declarative Components**: Decorator-based tool, resource, and prompt registration (60% less boilerplate vs. low-level SDK)
- **Provider System**: Local provider (decorators), filesystem provider (auto-discovery), MCP proxy provider, OpenAPI provider, skills provider, custom providers
- **Middleware**: Cross-cutting concerns (logging, rate limiting, authorization, caching, error handling, timing)
- **Built-in Patterns**: OAuth 2.1/OIDC proxy, dependency injection, authorization callables, lifecycle management (lifespans)
- **Composition**: Server mounting for modular architecture, namespace transforms for name isolation
- **Transformations**: Prompts-as-tools, resources-as-tools, versioning, visibility filtering
- **Interactive Apps**: Tool UIs rendered in conversations (Claude, ChatGPT, etc.)
- **Observability**: Native OpenTelemetry tracing, structured logging, Prometheus metrics
- **Configuration**: Project-level fastmcp.json for portable deployment
- **Escape Hatch**: Access to low-level MCP SDK for edge cases
- **Review Date**: 2026-06-01

1.7 5. Architecture Decisions

Per IEEE 42010 §5.7, architecture decisions and their rationale are captured in Architecture Decision Records.

ADR	Decision	Rationale	Viewpoints Affected	SRS Requirements
ADR-001	Python 3.11+ as implementation language	Native type hints, rich ecosystem, async/await support, enterprise adoption	Functional, Development	All FR-*
ADR-002	FastMCP v3.x as MCP framework	60% less boilerplate; provider system (local, filesystem, proxy, OpenAPI); built-in middleware, dependency injection, authorization; interactive Apps; composable via mounting; transformations (namespace, versioning); OpenTelemetry tracing; active maintenance	Functional, Development	FR-TOOL-*, FR-RSRC-*, FR-PROMPT-*
ADR-003	JWT/JWKS authentication	SSO integration, token lifecycle, proxy-compatible, cloud-native	Security	NFR-SEC-010–016
ADR-004	Stateless server design	Horizontal scaling, simple deployment, no session affinity	Deployment, Operational	NFR-PERF-010, DC-012
ADR-005	PostgreSQL + SQLite	ACID for audit, JSONB for schemas, SQLite for dev	Information, Development	FR-RSRC-003, NFR-PERF-008

ADR	Decision	Rationale	Viewpoints Affected	SRS Requirements
ADR-006	Streamable HTTP transport	Proxy-friendly, session management, SSE resumability, standard tooling	Functional, Deployment	FR-PROTO-001, FR-PROTO-025–031, DC-005
ADR-007	Distroless base images (Alpine optional)	**Distroless (primary)**: No shell, minimal attack surface (5-20MB), zero CVE exposure from OS packages, strongest compliance posture; **Alpine (optional)**: Development environments, complex builds needing apk/shell, 20-50MB with musl libc; **Decision criteria**: Production → distroless; dev/debug → Alpine	Security, Deployment	NFR-SEC-060–065, NFR-CNTR-001–012, DC-013

Per IEEE 42010 §5.6, correspondence rules define constraints between views and between the AD and SRS.

1.7.1 6.1 Cross-View Correspondences

Rule	Description	Views
CR-01	Every component in the Functional View that handles user requests must appear in the Security View with explicit trust boundary classification.	Functional Security
CR-02	Every component in the Functional View must have at least one metric or log point defined in the Operational View.	Functional Operational
CR-03	Every deployable unit in the Deployment View must map to a module in the Development View.	Deployment Development
CR-04	Every trust boundary in the Security View must have corresponding TLS configuration in the Deployment View.	Security Deployment
CR-05	Every health check in the Operational View must be implemented as a Kubernetes probe in the Deployment View.	Operational Deployment
CR-06	Every data entity in the Information View that contains PII must have masking rules defined in the Security View.	Information Security
CR-07	Every rate limit tier in the Operational View must correspond to a role in the Security View RBAC model.	Operational Security

1.7.2 6.2 AD-to-SRS Traceability

Every SRS requirement has at least one realizing element in this AD:

SRS Domain	AD Section
FR-PROTO-001–024	§4.1 Functional View (request flow, capability map)
FR-PROTO-025–034	§4.1.1 Layer Responsibilities (session management, lifecycle)
FR-RSRC-*	§4.1 Functional View, §4.2 Information View (data model)
FR-TOOL-*	§4.1 Functional View (capability map, layers)
FR-PROMPT-*	§4.1 Functional View (capability map)
FR-SAMP-*	§4.1 Functional View (capability map)
FR-ELIC-*	§4.1 Functional View (capability map)
FR-TASK-*	§4.1 Functional View (capability map)
FR-ORCH-*	§4.1.4 Multi-Server Orchestration
FR-GWWY-*	§4.1.5 AI Service Provider Gateway
NFR-SEC-001–072	§4.4 Security View (auth, RBAC, headers, validation, audit)
NFR-SEC-073–081	§4.4.4 Defense in Depth (MCP Protocol layer)

SRS Domain	AD Section
NFR-PERF-*	§4.5 Operational View (performance budget, resilience)
NFR-OBS-*	§4.5 Operational View (observability pipeline, metrics)
NFR-CNTR-*	§4.3 Deployment View (container spec, distribution)
DC-*	§4.3 Deployment View, §4.6 Development View

1.7.3 6.3 ADR Traceability

ADR	Realizing View Sections
ADR-001 (Python 3.11+)	§4.6 Development View (all implementation sections)
ADR-002 (FastMCP v3.x)	§4.1 Functional, §4.6.4 Framework Choice
ADR-003 (JWT/JWKS)	§4.4.2 OAuth 2.1 Flow
ADR-004 (Stateless)	§4.3.6 Scaling, §4.5.4 Performance Budget
ADR-005 (Database)	§4.2.1 Data Model
ADR-006 (Streamable HTTP)	§4.1.3 Request Flow, §4.1.1 Layer Responsibilities

---

1.8 7. Known Issues and Gaps

ID	Description	Status	Target	Resolution
GAP-01	Elicitation patterns documented with FastMCP v3 typed API	**Closed**	docs/03f-elicitation-patterns.md	v3.0.0: FastMCP v3 `ctx.elicit(message, response_type=Type)` with `Context` injection; typed result pattern matching (`AcceptedElicitation`/`DeclinedElic` response types: `str`, `int`, `bool`, `Literal`, `list` (multi-select), `dict` (titled options), `dataclass`, `BaseModel`; default values via `Field(default=...)`; multi-turn elicitation; confirmation (`response_type=None`); FR-ELIC-001–005 traced
GAP-02	Task patterns documented with full SRS traceability	**Closed**	docs/03g-task-patterns.md	v3.0.0: SRS reference corrected from FR-TASK-001–003 to FR-TASK-001–012; explicit traceability table mapping all 12 requirements to document sections (lifecycle states, task fields, progress notifications, cancel, list, TTL, taskSupport, session isolation, audit logging)

ID	Description	Status	Target	Resolution
GAP-03	Multi-server orchestration with FastMCP v3 composition	**Closed**	docs/03h-multi-server-orchestration.md	v2.0.0: FastMCP v3 Server Composition section with `mount()` (live link), `import_server()` (static copy), `create_proxy()` (remote HTTP, subprocess, npm/uvx); namespacing transforms (tools, resources, prompts); tag-based capability filtering (`include_tags`/`exclude_tags`); direct vs proxy mounting trade-offs; composition architecture diagram; FR-ORCH-001–005 traced
GAP-04	AI Service Provider Gateway patterns documented	**Closed**	docs/03i-ai-service-provider-gateway.md	v2.0.0: Provider-agnostic gateway architecture; endpoint configuration; rate limiting; enterprise headers; gateway client with connectivity verification; failover logic (429/5xx/timeout); credential rotation; MCP sampling integration; UAT test groups; FR-GWWY-001–012, NFR-SEC-058–065, CP-03 fully traced

ID	Description	Status	Target	Resolution
GAP-05	Protocol compatibility section rewritten with accurate MCP 2024-11-05 through 2025-11-25 coverage	**Closed**	docs/15-mcp-protocol-compatibility.md	v3.0.0: Full feature matrices (core, auth, metadata) for all 4 protocol versions; version negotiation with JSON-RPC initialize examples; capability negotiation reference; migration guidance for all upgrade paths; FastMCP v3 SDK-to-protocol mapping (ADR-002); transport selection guide
GAP-06	OAuth 2.1 patterns (PKCE, RFC 9728, OIDC, DCR) elaborated in security architecture	**Closed**	docs/02-security-architecture.md	v3.0.0: OAuth 2.1 + PKCE flow with sequence diagram; RFC 9728 Protected Resource Metadata endpoint; OIDC Discovery; Dynamic Client Registration; incremental scope consent; multi-provider support (JWT/JWKS, GitHub, Google, WorkOS); FastMCP v3 OAuth-Provider/JWTVerifier integration (ADR-002); NFR-SEC-001–009 traced

ID	Description	Status	Target	Resolution
GAP-07	Container security hardening (CIS, SBOM, signing, SLSA L3) addressed across deployment and security docs	**Closed**	docs/07-deployment-patterns.md, docs/02-security-architecture.md	07-deployment: CIS Docker Benchmark controls table, Cosign keyless signing, Syft SBOM (CycloneDX), SLSA provenance, Trivy scanning, hardened K8s SecurityContext; 02-security v3.0.0: SLSA L3 compliance table, multi-stage Dockerfile (Alpine → distroless per ADR-007), GitHub Actions security pipeline with safety/SBOM/Trivy/Cosign/SLSA; NFR-CNTR-001–012, NFR-CNTR-027–029 traced
GAP-08	Quantitative performance targets aligned to SRS NFR-PERF-001–023	**Closed**	docs/06a-performance-scalability.md	v2.0.0: Latency Budget table (9 metrics), Capacity Targets table (6 metrics), Resilience Targets table (7 patterns), all with SRS references; qualitative design requirements (NFR-PERF-010 stateless, NFR-PERF-018 health checks) traced to ADR-004 and Deployment View

1.9 Document Approval

Role	Name	Date
Author Enterprise Architect Security Lead	Mark Sigler	2026-02-24

Role	Name	Date
Engineering Lead		