Laozi by Pi.Tech

A domain-specific LLM insight generation engine for **healthcare metrics** with built-in hallucination prevention. Supports both **OpenAI** and **Azure OpenAI** (including secure deployments via **Azure Private Links**).

> ***Migrated from Go*** *— This is a complete rewrite of the laozi-go-plugin in idiomatic Java 21, keeping only the medical / healthcare example and adding Azure Private Link support.*

Key Features

- **Java 21** — Records, sealed interfaces, pattern matching, text blocks, virtual threads
- **Pre-computed comparisons** — Deterministic severity based on clinical thresholds
- **Validation + Regeneration** — Auto-retry on invalid LLM output (hallucination prevention)
- **Parallel execution** — Virtual thread executor bounded by configurable concurrency limit
- **Optional RAG** — Augment with medical reference documents
- **Dual provider support** — OpenAI API and Azure OpenAI (API key + Managed Identity)
- **Azure Private Link** — Secure architecture keeping all traffic within Azure backbone

---

Prerequisites

Requirement	Version
Java (JDK)	21+
Maven	3.9+
API Key	OpenAI or Azure OpenAI

Quick Start

1. Build

```
mvn clean package -DskipTests
```

2. Run with OpenAI

```
export LLM_PROVIDER=openai
export LLM_API_KEY=sk-...
java --enable-preview -jar target/laozi-medical-1.0.0.jar
```

3. Run with Azure OpenAI

```
export LLM_PROVIDER=azure
export AZURE_RESOURCE_NAME=my-openai-resource
export AZURE_DEPLOYMENT_NAME=gpt-4o-mini
export LLM_API_KEY=<azure-openai-key>
java --enable-preview -jar target/laozi-medical-1.0.0.jar
```

4. Run with Azure Private Link

```
export LLM_PROVIDER=azure-private
export AZURE_PRIVATE_ENDPOINT=my-openai.privatelink.openai.azure.com
export AZURE_DEPLOYMENT_NAME=gpt-4o-mini
export LLM_API_KEY=<key-or-managed-identity-token>
java --enable-preview -jar target/laozi-medical-1.0.0.jar
```

Programmatic Usage

```
import io.laozi.config.LaoziConfig;
import io.laozi.engine.LaoziEngine;
import io.laozi.llm.OpenAIClient;
import io.laozi.model.*;
import io.laozi.rag.InMemoryRAGStore;

// 1. Create engine
var engine = LaoziEngine.builder()
        .llm(new OpenAIClient(LaoziConfig.defaults(), System.getenv("LLM_API_KEY")))
        .rag(new InMemoryRAGStore())
        .context("patient", Map.of("age", 45))
        .build();

// 2. Add categories with clinical thresholds
```

```
engine.addCategory(new Category("glucose", "Blood Glucose", List.of(
        new Threshold("fasting_glucose", 70, 99, 70, 90,
                "mg/dL", "ADA",
                "https://diabetes.org/about-diabetes/diagnosis",
                "Normal fasting glucose range")
)));

// 3. Analyze
var metrics = Map.of("fasting_glucose", 105.0);
var insights = engine.analyzeAll(metrics);
```

Configuration

All settings are encapsulated in an immutable `LaoziConfig` record:

```
// Default configuration (mirrors original Go constants)
LaoziConfig config = LaoziConfig.defaults();

// Azure OpenAI
LaoziConfig config = LaoziConfig.azureOpenAI("my-resource", "gpt-4o-mini");

// Azure Private Link
LaoziConfig config = LaoziConfig.azurePrivateLink(
        "my-resource.privatelink.openai.azure.com",
        "gpt-4o-mini"
);
```

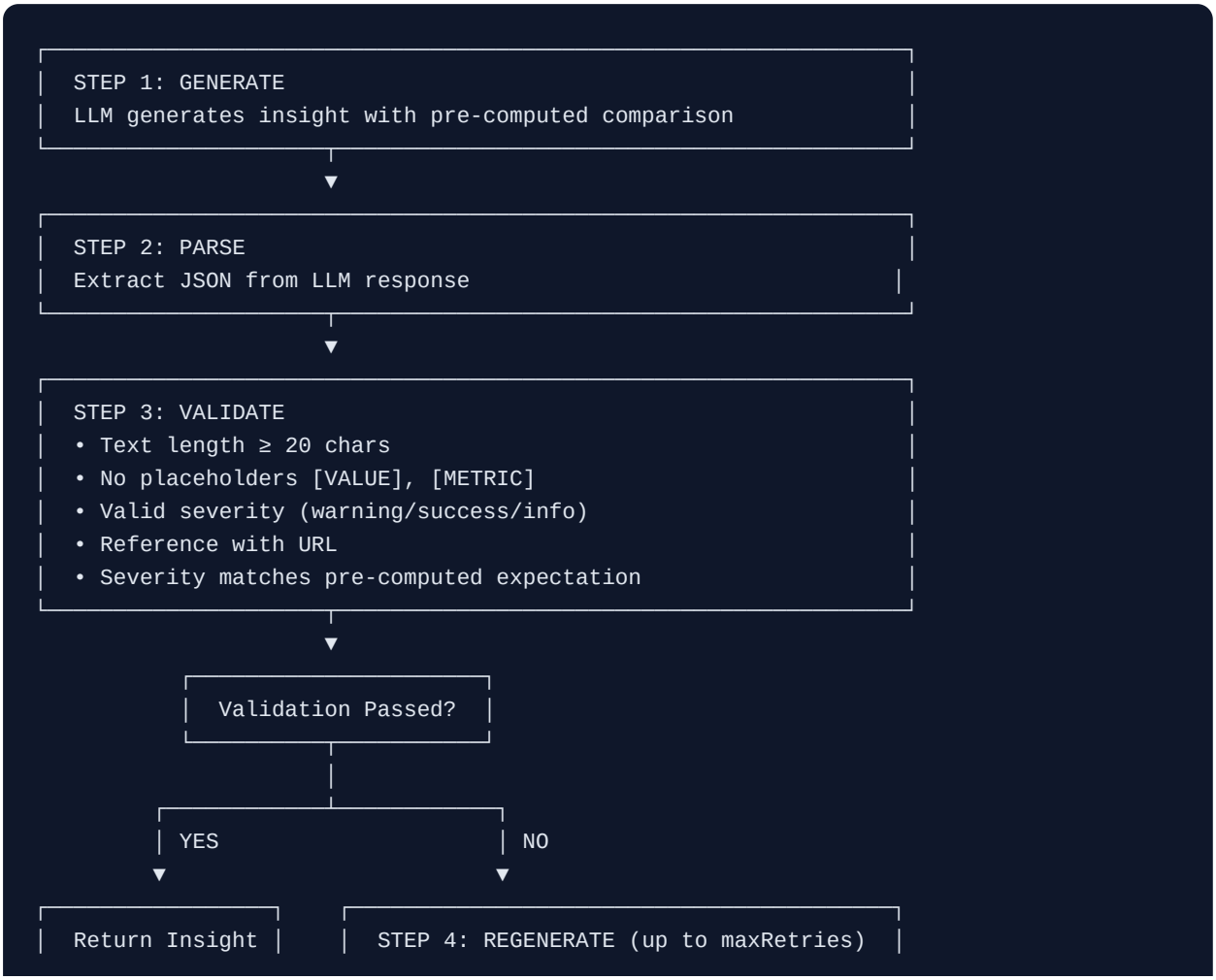
Configuration Parameters

Parameter	Default	Description
`llmModel`	`gpt-4o-mini`	Model name or deployment name
`llmEndpoint`	OpenAI URL	API endpoint URL
`llmTemperature`	`0.3`	Lower = more deterministic
`llmMaxTokens`	`500`	Max tokens per insight
`llmTopP`	`0.9`	Nucleus sampling
`maxParallelCalls`	`35`	Concurrent LLM requests (virtual threads)
`llmTimeout`	`30s`	Per-request timeout
`ragEnabled`	`true`	Enable/disable RAG

Parameter	Default	Description
`ragTopK`	`3`	Documents to retrieve
`ragMinSimilarity`	`0.7`	Cosine similarity threshold
`maxRetries`	`2`	Regeneration attempts
`minInsightTextLen`	`20`	Minimum insight text length
`requireReference`	`true`	Require source reference
`requireUrl`	`true`	Require URL in reference
`enforceSeverityMatch`	`true`	Severity must match pre-computed

Architecture

Insight Generation Pipeline



```
└──────────────────┘     │  Call LLM again with error message      │
                         └─────────────────────────────────────────┘
```

Anti-Hallucination: Pre-Computed Comparison

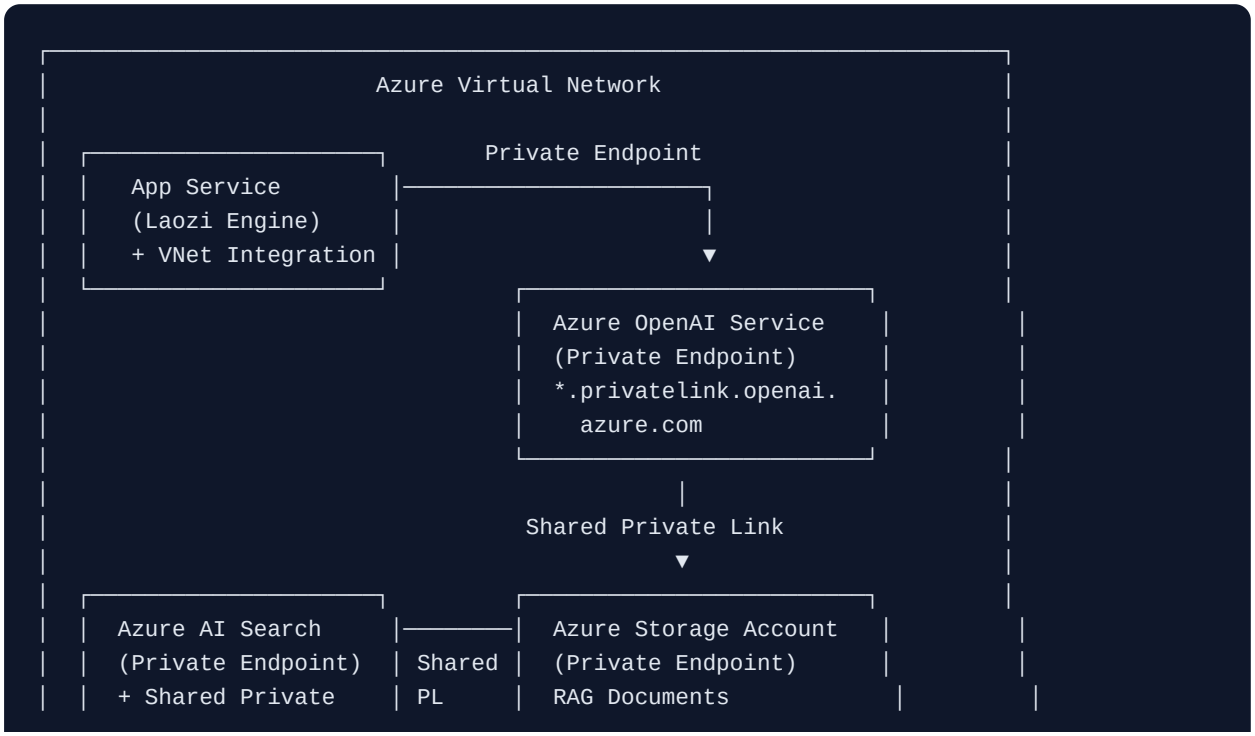
The key mechanism — the LLM's only job is to write prose around a deterministic result:

```
GUIDELINES:
📊 FASTING_GLUCOSE: Range 70.00 - 99.00 mg/dL
   Source: American Diabetes Association
   URL: https://diabetes.org/about-diabetes/diagnosis

PATIENT VALUES:
- fasting_glucose: 108.00 mg/dL

PRE-COMPUTED COMPARISON:
fasting_glucose: 108.00 > 99.00 (ABOVE maximum) → severity = warning

REQUIRED SEVERITY: warning
```

Azure Private Link Secure Architecture

This project is designed to integrate with the
[Azure AI Chatbot architecture with Private Links](#).

Architecture Overview



Key Components

Component	Purpose	Private Link DNS Zone
Azure OpenAI	LLM inference (GPT-4o)	`privatelink.openai.azure.com`
Azure AI Search	RAG document indexing & search	`privatelink.search.windows.net`
Azure Storage	RAG document storage	`privatelink.blob.core.windows.net`
App Service	Hosts the Laozi engine	VNet Integration (outbound)
Azure ML	Prompt Flow orchestration	`privatelink.api.azureml.ms`

Security Benefits

1. **No public internet exposure** — All Azure service communication uses private endpoints
2. **Azure backbone network** — Traffic stays within Microsoft's private network
3. **Managed Identity** — No API keys stored; uses Azure AD tokens
4. **Private DNS resolution** — Services resolve to private IPs within the VNet
5. **Fine-grained access control** — RBAC on each private endpoint connection

Setting Up Private Link

Step 1: Create Private Endpoints

For each Azure service (OpenAI, AI Search, Storage):

```
# Create Private Endpoint for Azure OpenAI
az network private-endpoint create \
    --name pe-openai \
    --resource-group myResourceGroup \
```

```
    --vnet-name myVNet \
    --subnet mySubnet \
    --private-connection-resource-id /subscriptions/.../Microsoft.CognitiveServices/accou
    --group-id account \
    --connection-name openai-connection
```

Step 2: Configure Private DNS Zones

```
# Create Private DNS Zone
az network private-dns zone create \
    --resource-group myResourceGroup \
    --name privatelink.openai.azure.com

# Link to VNet
az network private-dns zone vnet-link create \
    --resource-group myResourceGroup \
    --zone-name privatelink.openai.azure.com \
    --name openai-dns-link \
    --virtual-network myVNet \
    --registration-enabled false
```

Step 3: Create Shared Private Links (for AI Search)

```
# Shared Private Link from AI Search to Storage
az search shared-private-link-resource create \
    --name spl-storage \
    --service-name mySearchService \
    --resource-group myResourceGroup \
    --group-id blob \
    --resource-id /subscriptions/.../storageAccounts/myStorage \
    --request-message "AI Search needs access to RAG documents"
```

Step 4: Assign Managed Identity

```
# Assign Cognitive Services OpenAI User role
az role assignment create \
    --assignee <app-service-managed-identity-id> \
    --role "Cognitive Services OpenAI User" \
    --scope /subscriptions/.../Microsoft.CognitiveServices/accounts/myOpenAI
```

Step 5: Configure Java Application

```
// In your deployment, use Managed Identity:
var credential = new DefaultAzureCredentialBuilder().build();
var token = credential.getToken(new TokenRequestContext()
        .addScopes("https://cognitiveservices.azure.com/.default"))
```

```
        .block();

var config = LaoziConfig.azurePrivateLink(
        "my-openai.privatelink.openai.azure.com",
        "gpt-4o-mini");

var client = AzureOpenAIClient.withManagedIdentity(config, token.getToken());
```

Project Structure

```
laozi-java-medical/
├── pom.xml                              # Maven build (Java 21, shade plugin)
├── README.md                            # This file
├── docs/
│   └── DEPLOYMENT.md                    # Detailed deployment guide
├── src/
│   ├── main/
│   │   ├── java/io/laozi/
│   │   │   ├── config/
│   │   │   │   ├── LaoziConfig.java     # Immutable configuration record
│   │   │   │   └── PromptTemplates.java  # LLM prompt text blocks
│   │   │   ├── engine/
│   │   │   │   └── LaoziEngine.java     # Core pipeline engine
│   │   │   ├── llm/
│   │   │   │   ├── LLMClient.java       # Sealed interface
│   │   │   │   ├── LLMException.java    # Exception type
│   │   │   │   ├── OpenAIClient.java    # OpenAI implementation
│   │   │   │   └── AzureOpenAIClient.java # Azure OpenAI implementation
│   │   │   ├── model/
│   │   │   │   ├── Severity.java        # Enum with pattern matching
│   │   │   │   ├── Threshold.java       # Record
│   │   │   │   ├── Category.java        # Record
│   │   │   │   ├── Insight.java         # Record
│   │   │   │   └── RAGDocument.java     # Record
│   │   │   ├── rag/
│   │   │   │   ├── RAGStore.java        # Interface
│   │   │   │   └── InMemoryRAGStore.java # In-memory implementation
│   │   │   └── example/
│   │   │       └── HealthcareExample.java # Medical example (main class)
│   │   └── resources/
│   │       ├── application.properties    # Default config
│   │       ├── application-azure-private.yml  # Azure PL profile
│   │       └── logback.xml              # Logging config
│   └── test/
│       └── java/io/laozi/
│           └── LaoziEngineTest.java     # Unit tests
```

Java 21 Features Used

Feature	Usage
Records	`Threshold`, `Category`, `Insight`, `RAGDocument`, `LaoziConfig`
Sealed Interfaces	`LLMClient` permits `OpenAIClient`, `AzureOpenAIClient`
Pattern Matching	`switch` expressions in `Severity.fromString()`, provider factory
Text Blocks	All prompt templates in `PromptTemplates`
Virtual Threads	`Executors.newVirtualThreadPerTaskExecutor()` in `analyzeAll()`
String Templates	`.formatted()` throughout prompt building

Environment Variables

Variable	Required	Description
`LLM_PROVIDER`	No	`openai` (default), `azure`, `azure-private`
`LLM_API_KEY`	Yes	OpenAI or Azure OpenAI API key
`LLM_ENDPOINT`	No	Custom endpoint URL
`LLM_MODEL`	No	Model name (default: `gpt-4o-mini`)
`AZURE_RESOURCE_NAME`	Azure	Azure OpenAI resource name
`AZURE_DEPLOYMENT_NAME`	Azure	Azure deployment name
`AZURE_PRIVATE_ENDPOINT`	Azure PL	Private Link FQDN

License

MIT License