Raj's Onboarding ShowCase : 50 Days in (Nov 25 2025 - Jan 14 2026)

Raj Patil

January 13, 2026

Contents

1 Overview

showcase for projectnil : a rudimentary function as a service written in Java, employing wasm

2 Pnil

2.1 Tech Stack

2.1.1 Code

- Java 25, Spring Boot
- Chicory Wasm Runtime

2.1.2 DataBase

- Postgres 18

2.1.3 Message Queue

- pgmq 1.8

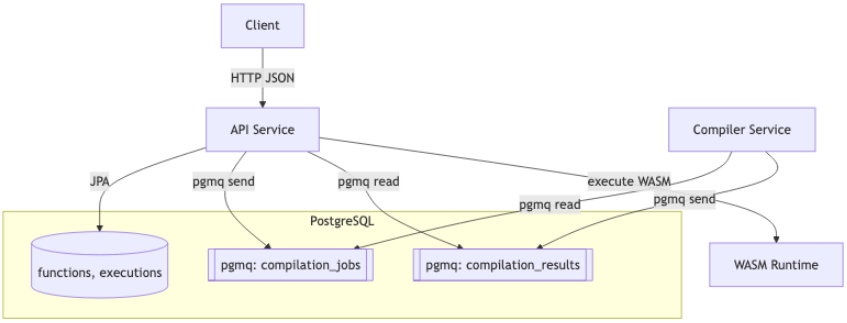
2.1.4 Migrations

- liquibase

2.1.5 Infra

- ansible
- podman
- scp'ing tarballs

3 Arch



4 Entities

- Functions
- Executions

5 User Stories

- CRD over Function
  - C on Function -> registering a function
  - R over Function -> polling to check status
- CRD over Execution
  - C on exec -> running a Function
  - R over Execution -> checking past runs

6 Appendix

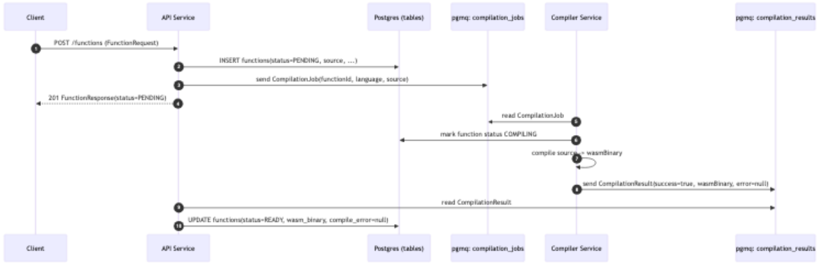
6.1 System Flows

6.1.1 Legend

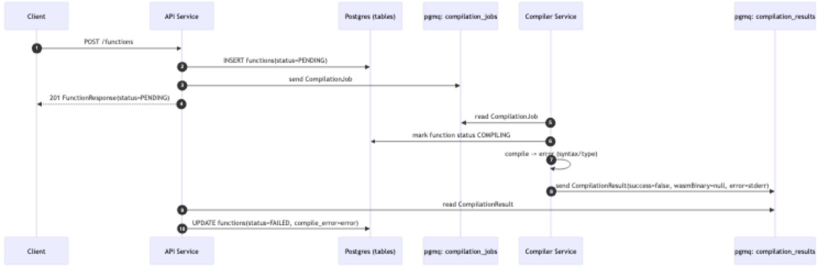
- API = API Service
- DB = PostgreSQL tables
- Q = pgmq queues

- CS = Compiler Service
- WR = WASM Runtime

6.1.2 Flow 1: Register Function (Compilation Succeeds)

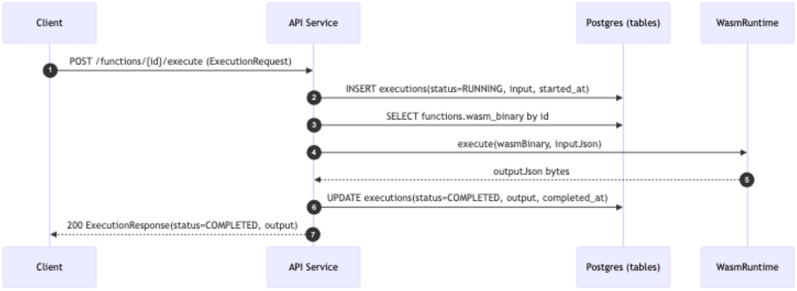

Client
API Service
Postgres (tables)
pgmq: compilation_jobs
Compiler Service
pgmq: compilation_results
1
POST /functions (FunctionRequest)
2
INSERT functions(status=PENDING, source, ...)
3
send CompilationJob(functionId, language, source)
4
201 FunctionResponse(status=PENDING)
5
read CompilationJob
6
mark function status COMPILING
7
compile source -> wasmBinary
8
send CompilationResult(success=true, wasmBinary, error=null)
9
read CompilationResult
10
UPDATE functions(status=READY, wasm_binary, compile_error=null)
Client
API Service
Postgres (tables)
pgmq: compilation_jobs
Compiler Service
pgmq: compilation_results

6.1.3 Flow 2: Register Function (Compilation Fails)


Client
API Service
Postgres (tables)
pgmq: compilation_jobs
Compiler Service
pgmq: compilation_results
1
POST /functions
2
INSERT functions(status=PENDING)
3
send CompilationJob
4
201 FunctionResponse(status=PENDING)
5
read CompilationJob
6
mark function status COMPILING
7
compile -> error (syntax/type)
8
send CompilationResult(success=false, wasmBinary=null, error=stderr)
9
read CompilationResult
10
UPDATE functions(status=FAILED, compile_error=error)
Client
API Service
Postgres (tables)
pgmq: compilation_jobs
Compiler Service
pgmq: compilation_results

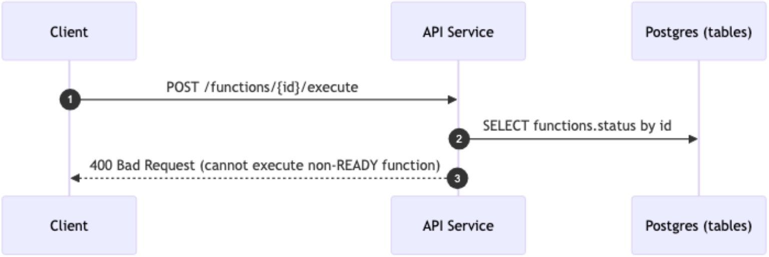
6.1.4 Flow 3: Execute Function (Success)

Precondition: ‘Function.status = READY‘ and ‘Function.wasmBinary ! null‘


Client
API Service
Postgres (tables)
WasmRuntime
1
POST /functions/{id}/execute (ExecutionRequest)
2
INSERT executions(status=RUNNING, input, started_at)
3
SELECT functions.wasm_binary by id
4
execute(wasmBinary, inputJson)
5
outputJson bytes
6
UPDATE executions(status=COMPLETED, output, completed_at)
7
200 ExecutionResponse(status=COMPLETED, output)
Client
API Service
Postgres (tables)
WasmRuntime

—

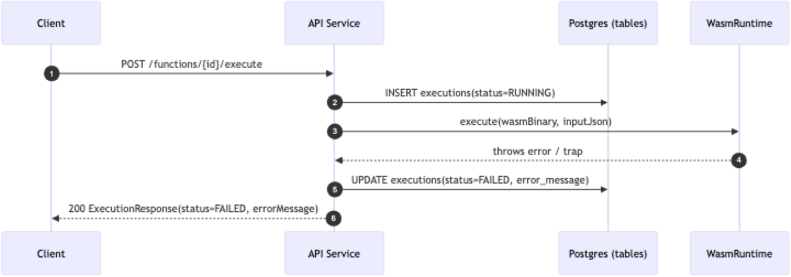
6.1.5 Flow 4: Execute Function (Rejected - Not Ready)


Client
API Service
Postgres (tables)
1
POST /functions/{id}/execute
2
SELECT functions.status by id
3
400 Bad Request (cannot execute non-READY function)
Client
API Service
Postgres (tables)

6.1.6 Flow 5: Execute Function (Runtime Failure)

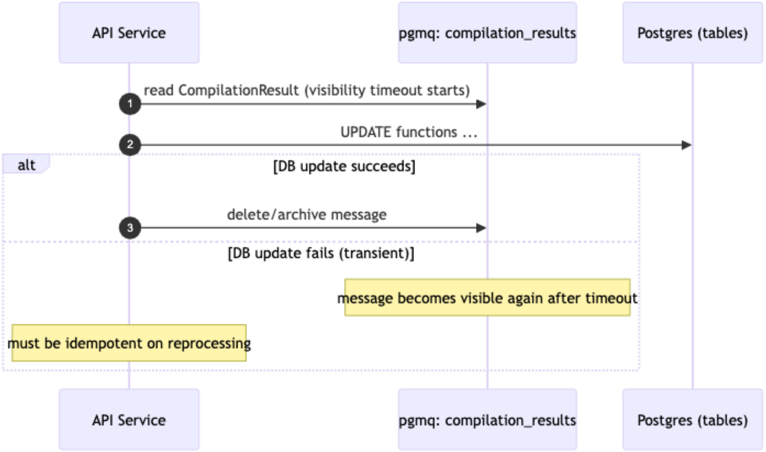
Examples of runtime failures:

- WASM trap (e.g., unreachable instruction)
- Invalid ABI export
- Invalid JSON encoding in output


Client
API Service
Postgres (tables)
WasmRuntime
1
POST /functions/{id}/execute
2
INSERT executions(status=RUNNING)
3
execute(wasmBinary, inputJson)
4
throws error / trap
5
UPDATE executions(status=FAILED, error_message)
6
200 ExecutionResponse(status=FAILED, errorMessage)
Client
API Service
Postgres (tables)
WasmRuntime

Note: Runtime failures return HTTP ‘200‘ because the platform operated correctly; the failure is in the user’s function.

6.1.7 Flow 6: Queue Redelivery / Retries


API Service
pgmq: compilation_results
Postgres (tables)
1
read CompilationResult (visibility timeout starts)
2
UPDATE functions ...
alt
[DB update succeeds]
3
delete/archive message
[DB update fails (transient)]
message becomes visible again after timeout
must be idempotent on reprocessing
API Service
pgmq: compilation_results
Postgres (tables)

6.1.8 Compiler Service Flow

1. Runtime Flow

   (a) **Startup**: Load configuration, establish pgmq connection, verify queues exist

   (b) **Job Consumption**: Poll ‘compilation$_{\text{jobs}}$‘ with visibility timeout

   (c) **Language Filtering**: Process only jobs matching supported language; archive others

   (d) **Compilation**:

   - Write source to temp file
   - Execute ‘asc <file> –binaryFile <output>.wasm –exportRuntime –runtime incremental –optimize‘
   - Capture stdout/stderr

   (e) **Result Publishing**:

   - Success: Base64-encode WASM, publish with ‘success=true‘
   - Failure: Publish with ‘success=false‘ and error message

   (f) **Cleanup**: Remove temp files, delete/archive queue message

2. Error Handling

Error Type	Action
Compilation failure (user code)	Publish ‘success=false‘, archive message
Transient infrastructure failure	Allow message to reappear via visibility timeout
Fatal error (DB unavailable)	Exit process, let orchestrator restart