

# Persistent Key-Value Store

Architectural & API Reference

Pujith Sai Kumar Korlepara

IIT Bombay (ID: 25M0787)

Email: [pujith@cse.iitb.ac.in](mailto:pujith@cse.iitb.ac.in) / [pujith22.sde@gmail.com](mailto:pujith22.sde@gmail.com)

GitHub: [pujith22](#) LinkedIn: [in/pujith22](#)

Portfolio: [cse.iitb.ac.in/ pujith](http://cse.iitb.ac.in/)

## 1 Postman API Collection

An interactive Postman collection is published for quick exploration and manual testing of the API surface. You can import it directly via the shared workspace URL below or by copying the JSON specification.

**Shared Collection URL:** [Postman Workspace Link](#)

The local development server (see `main_server.cpp`) binds to host `localhost` on port 2222. The collection above targets that base URL.

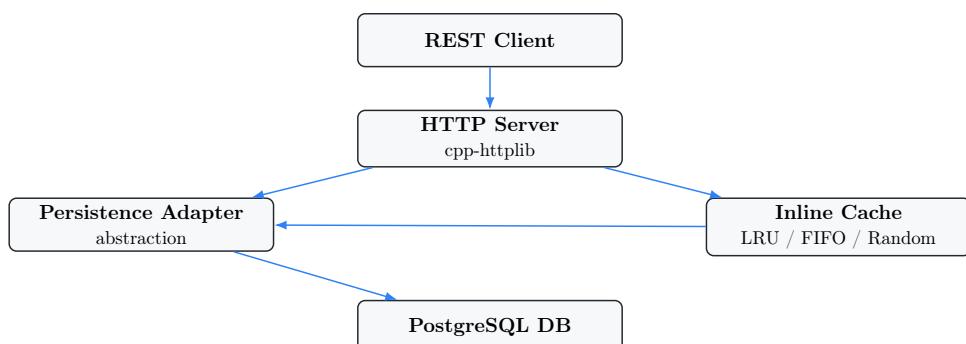
## Repository

GitHub: <https://github.com/pujith22/persistent-key-value-store>

## 2 Purpose

A C++17 HTTP service providing a persistent key-value cache backed by PostgreSQL. It exposes a JSON-centric REST API for CRUD and bulk operations while maintaining latency via an inline in-memory write-through cache. Startup fails fast if persistence cannot be reached, enforcing durability guarantees.

## 3 High-Level Architecture



**Figure 1:** Compact layered architecture: request flow across server, cache, and persistence.

## 4 Endpoint Catalog

The service exposes the following HTTP endpoints.

Method	Path	Description
<b>GET</b>	/	Machine-readable service catalog
<b>GET</b>	/home	Formatted documentation for available routes
<b>GET</b>	/get_key/:key_id	Return the value for the provided numeric key, caching it if not present in cache
<b>PATCH</b>	/bulk_query	Retrieve multiple keys in one request; missing keys noted in response; always returns success with errors appended
<b>POST</b>	/insert/:key/:value	Insert a key/value pair; conflicts return 409 with existing value; write-through to cache and persistence
<b>POST</b>	/bulk_update	Transactional pipeline for create/get/insert/update; rolls back on failure and returns failure response
<b>DELETE</b>	/delete_key/:key	Remove the provided key from both cache and persistence layer
<b>PUT</b>	/update_key/:key/:value	Update an existing key with a new value in both cache and persistence layer
<b>GET</b>	/health	Report service health and uptime
<b>GET</b>	/metrics	Expose cache metrics including hit/miss counts
<b>GET</b>	/stop	Gracefully stop the server (testing/debug only; not for production)

## 5 Screenshots

```

Pretty-print ✘
{
  "description": "HTTP-accessible cache with inline persistence adapter hooks",
  "links": [
    {
      "name": "/health",
      "path": "/home",
      "metrics": "/metrics"
    }
  ],
  "routes": [
    {
      "description": "Machine-readable service catalog",
      "method": "GET",
      "path": "/"
    },
    {
      "description": "Formatted documentation for available routes",
      "method": "GET",
      "path": "/home"
    },
    {
      "description": "Return the value for the provided numeric key caching it if not present in cache",
      "method": "GET",
      "path": "/get_key/:key_id"
    },
    {
      "description": "Retrieve multiple keys in one request; missing keys noted in response, always return success response with error appended to the response",
      "method": "PATCH",
      "path": "/bulk_query"
    },
    {
      "description": "Insert a key/value pair; conflicts return 409 with existing value, writes both to cache and persistence layer (note that we are using write-through type of cache)",
      "method": "POST",
      "path": "/insert/:key/:value"
    },
    {
      "description": "Transactional Commit pipeline for create/get/insert/update operations, rollbacks in case of failure and returns failure response",
      "method": "POST",
      "path": "/bulk_update"
    },
    {
      "description": "Remove the provided key from both the cache and persistence layer",
      "method": "DELETE",
      "path": "/delete_key/:key"
    },
    {
      "description": "Update an existing key with a new value to both the cache and persistence layer",
      "method": "PUT",
      "path": "/update_key/:key/:value"
    },
    {
      "description": "Report service health and uptime",
      "method": "GET",
      "path": "/health"
    },
    {
      "description": "Expose cache metrics including hit/miss counts",
      "method": "GET",
      "path": "/metrics"
    },
    {
      "description": "Gracefully stop the server (testing/debug only), shouldn't be available in prod environment",
      "method": "GET",
      "path": "/stop"
    }
  ],
  "service": "Persistent Key Value Store",
  "version": "1.0"
}

```

(a) Root endpoint (service catalog)

PERSISTENT KEY VALUE STORE  
Fast HTTP interface for managing key/value pairs with persistence layer in the backend (Postgres)

[View JSON Service Catalog](#)

METHOD	ROUTE	DESCRIPTION
GET	/	Machine-readable service catalog
GET	/home	Formatted documentation for available routes
GET	/get_key/:key_id	Return the cached value for the provided numeric key
POST	/bulk_query	Retrieve multiple keys in one request; missing keys noted in response
POST	/insert/:key/:value	Insert a key/value pair; conflicts return 409 with existing value
PATCH	/bulk_update	Partial commit pipeline for insert/update operations
DELETE	/delete_key/:key	Remove the provided key from the cache
PUT	/update_key/:key/:value	Update an existing key with a new value
GET	/health	Report service health and uptime
GET	/metrics	Expose cache metrics including hit/miss counts
GET	/stop	Gracefully stop the server (testing/debug only)

Copyright © Pujith Sai Kumar Korlapara. All rights reserved.

(b) Home HTML endpoint

You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.

History New Import

POST http://localhost:2222/get\_key/250

GET http://localhost:2222/get\_key/22

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "found": true,
3   "query_key": "22",
4   "value": "pujith"
5 }
```

Status: 200 OK Time: 2 ms Size: 151 B Save Response

Create collections in Postman  
Use collections to save your requests and share them with others.

(a) Cache hit on get\_key

You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.

History New Import

POST http://localhost:2222/get\_key/250

GET http://localhost:2222/get\_key/2

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "cache_populated": true,
3   "found": true,
4   "query_key": "2",
5   "source": "persistence",
6   "value": "bar"
7 }
```

Status: 200 OK Time: 4 ms Size: 193 B Save Response

Create collections in Postman  
Use collections to save your requests and share them with others.

(b) Hydrating uncached key from persistence

The screenshot shows the Postman interface with a failed API call. The URL is `http://localhost:2222/get_key/250`. The response status is 404 Not Found, with a message: "Status: 404 Not Found Time: 3 ms Size: 209 B Save Response". The response body is a JSON object:

```

1
2   + "found": false,
3   + "persistence_checked": true,
4   + "query_key": "22",
5   + "reason": "key not present in cache or persistence"
6

```

(a) Key not found scenario

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:2222/insert/22/pujith`. The response status is 201 Created, with a message: "Status: 201 Created Time: 7 ms Size: 169 B Save Response". The response body is a JSON object:

```

1
2   + "created": true,
3   + "key": "22",
4   + "persisted": true,
5   + "value": "pujith"
6

```

(b) Successful clean insert

The screenshot shows the Postman interface with a request to `http://localhost:2222/insert/22/pujith`. The response status is 409 Conflict, with a response body containing:

```

1
2   "error": "key exists",
3   "existing_value": "pujith",
4   "key": "22"
5   "reason": "insert rejected because key already exists",
6   "value": "pujith"
7

```

(a) Insert conflict (409)

The screenshot shows the Postman interface with a request to `http://localhost:2222/update_key/250/buffalo`. The response status is 200 OK, with a response body containing:

```

1
2   "key": "250",
3   "persisted": true,
4   "persistence_checked": true,
5   "updated": true,
6   "value": "elonmusk"
7

```

(b) Successful update operation

The screenshot shows the Postman interface with a history panel on the left listing various API calls. In the main center, a PUT request is being prepared to update a key. The URL is `http://localhost:2222/update_key/250/buffalo`. The body contains the JSON object `{"key": "elonmusk"}`. The response tab shows a 404 Not Found status with the message: "Key not found".

(a) Update when key absent (404)

The screenshot shows the Postman interface with a history panel on the left. In the main center, a DELETE request is being prepared to delete a key. The URL is `http://localhost:2222/delete_key/22`. The response tab shows a 204 No Content status.

(b) Successful deletion (204)

The screenshot shows a Postman interface with a history panel on the left containing various API requests. In the main workspace, a DELETE request is being made to `http://localhost:2222/delete_key/10`. The request body is empty. The response status is 404 Not Found, and the response body contains the JSON object:

```

1 "error": "not found",
2 "key": "10",
3 "persistence_checked": true,
4 "reason": "key not present in cache or persistence"
5
6

```

(a) Deletion when key not found

The screenshot shows a Postman interface with a history panel on the left. In the main workspace, a PATCH request is being made to `http://localhost:2222/bulk_query`. The request body is a JSON object with an array of data points:

```

1 "data": [
2   250,
3   10,
4   350,
5   1,
6   10,
7   "invalid type",
8   22,
9   2
10 ]
11
12

```

The response status is 200 OK, and the response body is a large JSON object containing multiple entries, each representing a key and its status (e.g., found, not found, hit persistence, miss). The JSON structure is too large to fully display here.

(b) Bulk query results with mixed statuses

The screenshot shows the Postman interface with a successful response from a POST request to `http://localhost:2222/bulk_update`. The response body is a JSON object containing an array of operations and their results. The operations include insert, update, get, and delete for keys 25, 1, and 26, with values like 'elephant', 'Lion', and 'Tiger'. The status is 'ok' for most operations.

```

{
  "operations": [
    {
      "operation": "insert",
      "key": 25,
      "value": "elephant"
    },
    {
      "operation": "update",
      "key": 1,
      "value": "Lion"
    },
    {
      "operation": "get",
      "key": 1
    },
    {
      "operation": "delete",
      "key": 25
    }
  ]
}

```

(a) Bulk update transaction success report

The screenshot shows the Postman interface with a successful response from a POST request to `http://localhost:2222/bulk_update`. The response body is a JSON object indicating a failure at index 4 due to a key not present error. The status is 'failed' for the update operation at index 4.

```

{
  "operations": [
    {
      "operation": "insert",
      "key": 25,
      "value": "elephant"
    },
    {
      "operation": "update",
      "key": 1,
      "value": "Lion"
    },
    {
      "operation": "get",
      "key": 1
    },
    {
      "operation": "delete",
      "key": 25
    },
    {
      "operation": "update",
      "key": 25,
      "value": "Tiger"
    }
  ]
}

```

(b) Bulk update transaction failure and rollback

The screenshot shows the Postman interface with a request to `http://localhost:2222/metrics`. The response status is 200 OK, time is 3 ms, and size is 163 B. The response body is a JSON object:

```

{
  "bytes": 245,
  "entries": 4,
  "evictions": 0,
  "hits": 9,
  "misses": 15
}

```

(a) Metrics endpoint showing cache statistics

The screenshot shows the Postman interface with a request to `http://localhost:2222/health`. The response status is 200 OK, time is 2 ms, and size is 158 B. The response body is a JSON object:

```

{
  "status": "ok",
  "uptime_ms": 1207725
}

```

(b) Health endpoint with uptime

The screenshot shows a Postman request to `http://localhost:2222/get_key/25`. The 'Params' tab is selected, showing a query parameter named 'Key' with the value 'stopping'. The 'Body' tab shows the response body:

```

1  {
2     "stopping": true
3 }

```

(a) Stop endpoint (dev only)

```

○ → persistent-key-value-store git:(feature/upgrade-system-to-use-http-server) ✘ ./kv_server.out --json-logs
{"cache_policy":"LRU","db_connection_status":"ok","json_logging_enabled":true,"listen":{"host":"localhost","port":2222},"ready":true,"start_time_ms":1762528983299,"type":"start-up"}
{"body_bytes":131,"method":"PATCH","path":"/bulk_update","path_param_count":0,"type":"request"}
{"body_bytes":109,"content_type":"application/json","duration_ms":200,"status":200,"type":"response"}
{"body_bytes":397,"method":"POST","path":"/bulk_update","path_param_count":0,"type":"request"}
{"body_bytes":608,"content_type":"application/json","duration_ms":3,"reason":"ok","status":200,"type":"response"}
{"body_bytes":475,"method":"POST","path":"/bulk_update","path_param_count":0,"type":"request"}
{"body_bytes":505,"method":"POST","path":"/bulk_update","path_param_count":0,"type":"request"}
{"body_bytes":78,"content_type":"application/json","duration_ms":3,"reason":"ok","status":200,"type":"response"}
{"body_bytes":78,"content_type":"application/json","duration_ms":3,"reason":"ok","status":200,"type":"response"}
{"body_bytes":385,"method":"POST","path":"/bulk_update","path_param_count":0,"type":"request"}
{"body_bytes":109,"content_type":"application/json","duration_ms":1,"reason":"ok","status":200,"type":"response"}
{"body_bytes":98,"method":"GET","path":"/get_key/25","path_param_count":1,"path_params":{"key_id":25}, "type":"request"}
{"body_bytes":47,"content_type":"application/json","duration_ms":8,"reason":"ok","status":200,"type":"response"}
{"body_bytes":406,"method":"POST","path":"/bulk_update","path_param_count":0,"type":"request"}
{"body_bytes":397,"content_type":"application/json","duration_ms":0,"reason":"ok","status":200,"type":"response"}
{"body_bytes":109,"content_type":"DELETE","path":"/delete_key/25","path_param_count":1,"path_params":{"key_id":25}, "type":"request"}
{"body_bytes":109,"content_type":"application/json","duration_ms":1,"reason":"ok","status":204,"type":"response"}
{"body_bytes":22,"method":"DELETE","path":"/delete_key/25","path_param_count":1,"path_params":{"key_id":25}, "type":"request"}
{"body_bytes":20,"content_type":"application/json","duration_ms":0,"reason":"deleted","status":204,"type":"response"}
{"body_bytes":22,"method":"DELETE","path":"/delete_key/2532","path_param_count":1,"path_params":{"key_id":2532}, "type":"request"}
{"body_bytes":112,"content_type":"application/json","duration_ms":0,"reason":"not found","reason_detail":"key not present in cache or persistence", "status":404,"type":"response"}
{"body_bytes":22,"method":"DELETE","path":"/delete_key/27","path_param_count":1,"path_params":{"key_id":27}, "type":"request"}
{"body_bytes":109,"content_type":"application/json","duration_ms":0,"reason":"not found","reason_detail":"key not present in cache or persistence", "status":404,"type":"response"}
{"body_bytes":109,"content_type":"DELETE","path":"/path_param_count/22","path_param_count":22,"path_params":{"key":22}, "type":"request"}
{"body_bytes":109,"content_type":"application/json","duration_ms":0,"reason":"not found","reason_detail":"key not present in cache or persistence", "status":404,"type":"response"}
{"body_bytes":8,"method":"GET","path":"/get_key/27","path_param_count":1,"path_params":{"key_id":27}, "type":"request"}
{"body_bytes":109,"content_type":"application/json","duration_ms":0,"reason":"not found","reason_detail":"key not present in cache or persistence", "status":404,"type":"response"}
{"body_bytes":109,"content_type":"POST","path":"/insert/22/pujith","path_param_count":2,"path_params":{"key":22,"value": "pujith"}, "type":"request"}
{"body_bytes":61,"content_type":"application/json","duration_ms":1,"reason":"created","status":201,"type":"response"}
{"body_bytes":109,"method":"POST","path":"/insert/1/foo","path_param_count":2,"path_params":{"key":1,"value": "foo"}, "type":"request"}
{"body_bytes":124,"content_type":"application/json","duration_ms":0,"reason":"conflict_key_exists","reason_detail":"Insert rejected because key already exists", "status":409,"type":"response"}
{"body_bytes":8,"method":"PUT","path":"/update/1/bar","path_param_count":2,"path_params":{"key":2,"value": "bar"}, "type":"request"}
{"body_bytes":57,"content_type":"application/json","duration_ms":0,"reason":"created","status":201,"type":"response"}
{"body_bytes":22,"method":"PUT","path":"/update_key/250/elonmusk","path_param_count":2,"path_params":{"key":250,"value": "elonmusk"}, "type":"request"}
{"body_bytes":91,"content_type":"application/json","duration_ms":1,"reason":"updated","status":200,"type":"response"}
{"body_bytes":22,"method":"PUT","path":"/update_key/250/elonmusk","path_param_count":2,"path_params":{"key":250,"value": "elonmusk"}, "type":"request"}
{"body_bytes":91,"content_type":"application/json","duration_ms":0,"reason":"updated","status":200,"type":"response"}
{"body_bytes":133,"content_type":"application/json","duration_ms":0,"reason":"not found","reason_detail":"key not present in cache or persistence", "status":404,"type":"response"}
{"body_bytes":22,"method":"PUT","path":"/update_key/250/elonmusk","path_param_count":2,"path_params":{"key":250,"value": "elonmusk"}, "type":"request"}
{"body_bytes":22,"method":"PUT","path":"/update_key/250/elephant","path_param_count":2,"path_params":{"key":250,"value": "elephant"}, "type":"request"}
{"body_bytes":8,"method":"GET","path":"/get_key/250/elonmusk","path_param_count":1,"path_params":{"key":250}, "type":"request"}
{"body_bytes":8,"method":"GET","path":"/get_key/250/elephant","path_param_count":1,"path_params":{"key":250}, "type":"request"}
{"body_bytes":35,"content_type":"application/json","duration_ms":0,"reason":"ok","status":200,"type":"response"}
{"body_bytes":0,"method":"GET","path":"/metrics","path_param_count":0,"type":"request"}
{"body_bytes":60,"content_type":"application/json","duration_ms":0,"reason":"ok","status":200,"type":"response"}

```

(b) Structured JSON logging output sample

```

* persistent-key-value-store git:(feature/upgrade-system-to-use-http-server) ✘ ./kv_server.out --policy=FIFO --json-logs
{"cache_policy": "FIFO", "db_connection_status": "ok", "json_logging_enabled": true, "listen": {"host": "localhost", "port": 2222}, "ready": true, "start_time_ms": 1762530921896, "type": "startup"}
{"body_bytes": 565, "method": "POST", "path": "/bulk_update", "path_param_count": 0, "type": "request"}
{"bytes": 777, "content_type": "application/json", "duration_ms": 6, "reason": "ok", "reason_detail": "key not present", "status": 200, "type": "response"}
{"body_bytes": 0, "method": "GET", "path": "/get_key/25", "path_param_count": 1, "path_params": {"key_id": "25"}, "type": "request"}
{"bytes": 110, "content_type": "application/json", "duration_ms": 0, "reason": "not_found", "reason_detail": "key not present in cache or persistence", "status": 404, "type": "response"}
{"body_bytes": 598, "method": "POST", "path": "/bulk_update", "path_param_count": 0, "type": "request"}
{"bytes": 360, "content_type": "application/json", "duration_ms": 1, "reason": "ok", "reason_detail": "key not present", "status": 200, "type": "response"}
{"body_bytes": 0, "method": "GET", "path": "/get_key/25", "path_param_count": 1, "path_params": {"key_id": "25"}, "type": "request"}
{"bytes": 140, "content_type": "application/json", "duration_ms": 0, "reason": "not_found", "reason_detail": "key not present in cache or persistence", "status": 404, "type": "response"}
{"body_bytes": 0, "method": "POST", "path": "/insert/elephant", "path_param_count": 2, "path_params": {"key": "25", "value": "elephant"}, "type": "request"}
{"bytes": 63, "content_type": "application/json", "duration_ms": 1, "reason": "created", "status": 201, "type": "response"}
{"body_bytes": 0, "method": "GET", "path": "/get_key/25", "path_param_count": 1, "path_params": {"key_id": "25"}, "type": "request"}
{"body_bytes": 0, "method": "GET", "path": "/get_key/25", "path_param_count": 1, "path_params": {"key_id": "25"}, "type": "response"}
{"body_bytes": 50, "content_type": "application/json", "duration_ms": 0, "reason": "ok", "status": 200, "type": "response"}
{"body_bytes": 598, "method": "POST", "path": "/bulk_update", "path_param_count": 0, "type": "request"}
{"bytes": 780, "content_type": "application/json", "duration_ms": 2, "reason": "ok", "reason_detail": "key not present", "status": 200, "type": "response"}
{"body_bytes": 0, "method": "GET", "path": "/get_key/25", "path_param_count": 1, "path_params": {"key_id": "25"}, "type": "request"}
{"bytes": 150, "content_type": "application/json", "duration_ms": 0, "reason": "ok", "status": 200, "type": "response"}
{"body_bytes": 0, "method": "POST", "path": "/bulk_update", "path_param_count": 0, "type": "request"}
{"bytes": 780, "content_type": "application/json", "duration_ms": 3, "reason": "ok", "reason_detail": "key not present", "status": 200, "type": "response"}
{"body_bytes": 400, "method": "POST", "path": "/bulk_update", "path_param_count": 0, "type": "request"}
{"bytes": 610, "content_type": "application/json", "duration_ms": 3, "reason": "ok", "status": 200, "type": "response"}
{"body_bytes": 0, "method": "GET", "path": "/get_key/25", "path_param_count": 1, "path_params": {"key_id": "25"}, "type": "request"}
{"body_bytes": 0, "method": "GET", "path": "/stop", "path_param_count": 0, "type": "request"}
{"bytes": 17, "content_type": "application/json", "duration_ms": 0, "reason": "ok", "status": 200, "type": "response"}

```

**Figure 11:** Server startup with custom cache eviction policy.

## 6 Credits

Third-party libraries: [nlohmann/json](#) and [cpp-httplib](#). Licensed under MIT (see project LICENSE).