

Developers Conference 2026

MAKING AN API FOR BUILDING APIS

JULY

2026

**From First Line to Production: The Technical Journey
Behind Building a Complete API Platform**

CONTENTS

01. Who am I?

02. What am I trying to solve?

03. What is Kinesis API?

04. Technology Stack

05. Why Rust?

06. Development Timeline

07. Foundations

08. Foundations: Mappings

09. API & Logic Explosion

10. Stabilization & UX

11. Productization & Scale

12. Multi-DB & Analytics

13. Next steps



Founder & Lead Developer - Kinesis API

Kishan Takoordyal

Professionally, I act as the Infrastructure and Support Team Leader at **Esokia**. I also develop games, APIs and websites during my free time.

I have authored a book on Unity, and have participated in numerous contests while contributing to Open Source.

"Nothing is Impossible" - Unknown

WHAT I AM TRYING TO SOLVE?

Modern API development is fragmented and complex:

Multiple Tools

Separate systems for API logic, databases, CMS, user management, localization...

Slow Iteration

Backend changes require code compilation, testing, and redeployment

External Dependencies

Reliance on third-party databases creates deployment complexity and failure points

Steep Learning Curve

Teams need expertise across multiple technologies and frameworks

Configuration Hell

Coordinating different services, versions, and configurations

WHAT IS KINESIS API?

An all-in-one framework for building, managing, and scaling APIs

X Engine

Build complete APIs without writing backend code through an intuitive flow-based system. Design custom workflows using drag-and-drop blocks, processors, and conditional logic. Create complex data transformations, validation rules, and API integrations visually—then deploy instantly. Perfect for rapid prototyping and production systems alike.

Kinesis DB

A custom-built, high-performance ACID-compliant database engine developed from scratch in Rust. Features multiple storage engines (In-Memory, On-Disk, Hybrid), memory-mapped I/O for ultra-fast operations, and write-ahead logging for crash recovery. Zero external database dependencies means simplified deployment and reduced failure points.

Complete Tooling

Production-ready capabilities built-in: full content management system with Markdown blogging, multi-language localization with variable interpolation, role-based user authentication with 2FA/OTP, media management, automated backups with compression, version control for all content, advanced search with custom query language, component analytics, ticketing system with email notifications, and interactive REPL for real-time testing.

Technology Stack

Backend

Built for Performance & Safety

Core Framework:

- Rust - Memory-safe systems programming language
- Rocket - Type-safe, ergonomic web framework
- Tera - Powerful templating engine (Jinja2-inspired)

Database & Storage:

- Kinesis DB - Custom ACID-compliant embedded database
- Diesel ORM - Type-safe SQL query builder (for external DB support)
- Memory-mapped I/O - Ultra-fast data access

API & Integration:

- utoipa - OpenAPI documentation generation
- Lettre - Email delivery system
- MaxMind GeoIP2 - Geographic IP lookup
- Serde - High-performance serialization

Infrastructure:

- Tokio - Async runtime for concurrent operations
- Cryptography libraries - bcrypt, JWT, 2FA/OTP
- Compression - gzip, tar for backups

Frontend

Modern, Minimal, Fast

Styling & UI:

- Tailwind CSS - Utility-first CSS framework
- DaisyUI - Component library
- Remix Icon - Open-source icon set
- Jost & Montserrat - Clean, professional typography

Content & Rendering:

- Markdown-it - Extensible Markdown parser
- PrismJS - Lightweight syntax highlighting
- MermaidJS - Diagram and flowchart generation

Data Visualization:

- Chart.js - Interactive charts with Luxon adapter
- Leaflet - Map visualization
- JMESPath - JSON query language

WHY RUST?

SPEED

Zero-cost abstractions: No runtime overhead

Compiled to native code:
Blazing fast execution

Memory-mapped I/O and
direct system access

CONCURRENCY

Fearless concurrency
through ownership system

Tokio async runtime for
handling thousands of
connections

Thread-safe by default

SAFETY

Memory safety without
garbage collection

No null pointer exceptions,
no data races

Compile-time error
detection prevents runtime
crashes

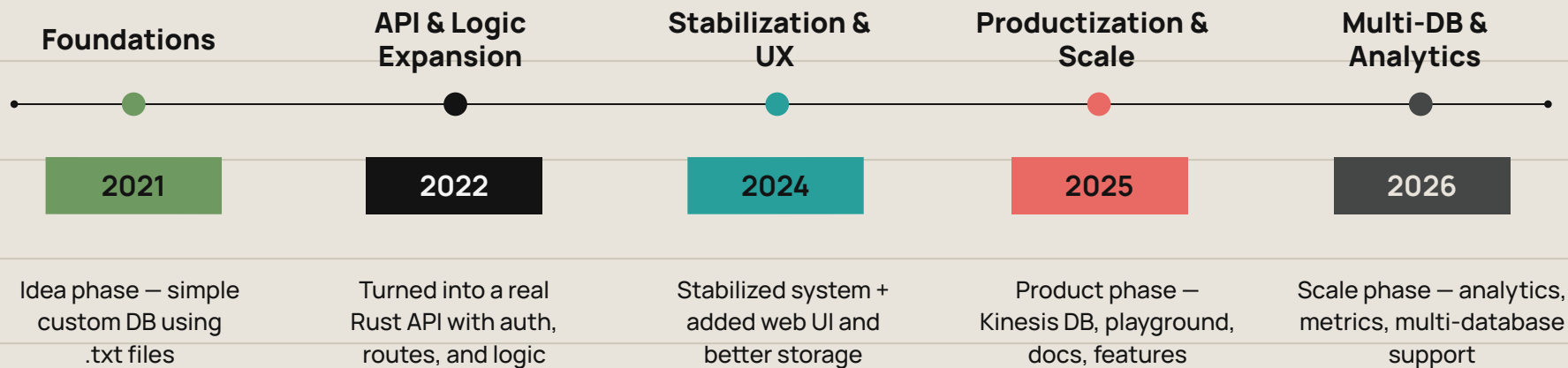
REFACTORING

Compiler catches breaking
changes before runtime

Type system ensures
correctness across the
entire codebase

Refactor with
confidence—no hidden bugs
or memory issues

DEVELOPMENT TIMELINE





FOUNDATIONS

2021

Foundations: Core Components

Initially, the project was just to be a specialized & fast database in Rust and the API layer was meant to be coded in a framework on top of NodeJS.

The first core components of the project were created to represent data.

```
5  #[derive(Default, Debug, Clone)]
6  pub struct Mapping {
7      id: String,
8      file_name: String,
9  }
```

```
3  #[derive(Default, Debug, Clone)]
4  pub struct Config {
5      pub name: String,
6      pub value: String,
7  }
```

```
4  #[derive(Default, Debug, Clone)]
5  pub struct CustomStructure {
6      pub id: String,
7      name: String,
8      structures: Vec<Structure>,
9  }
```

```
6  #[derive(Debug, Clone)]
7  pub enum Role {
8      ROOT,
9      ADMIN,
10     AUTHOR,
11 }
12
13 impl Default for Role {
14     fn default() -> Self {
15         Role::AUTHOR
16     }
17 }
18
19 #[derive(Default, Debug, Clone)]
20 pub struct User {
21     pub id: String,
22     first_name: String,
23     last_name: String,
24     username: String,
25     email: String,
26     password: String,
27     role: Role,
28 }
```

```
3  #[derive(Default, Debug, Clone)]
4  pub struct Project {
5      id: String,
6      name: String,
7      description: String,
8      api_path: String,
9  }
```

```
6  #[derive(Default, Debug, Clone)]
7  pub struct Collection {
8      id: String,
9      project_id: String,
10     name: String,
11     description: String,
12     structures: Vec<Structure>,
13     custom_structures: Vec<CustomStructure>,
14 }
```

```
5  #[derive(Default, Clone, Debug)]
6  pub struct EncryptionKey(pub String);
```

```
3  #[derive(Debug, Clone)]
4  pub enum Type {
5      TEXT,
6      EMAIL,
7      PASSWORD,
8      RICHTEXT,
9      NUMBER,
10     ENUM,
11     DATE,
12     MEDIA,
13     BOOLEAN,
14     UID,
15     JSON,
16     CUSTOM(String),
17 }
18
19 impl Default for Type {
20     fn default() -> Self {
21         Type::TEXT
22     }
23 }
24
25 #[derive(Default, Debug, Clone)]
26 pub struct Structure {
27     pub id: String,
28     name: String,
29     stype: Type,
30     default: String,
31     min: usize,
32     max: usize,
33     encrypted: bool,
34     unique: bool,
35     regex_pattern: String,
36     array: bool,
37 }
```

Foundations: Mappings

Mappings were a special type of component that served as a K/V store representation of components and the .txt file that was used for data storage.

All Components had logic to add separators between each field and an associated `fetch_all_xyz` and `save_all_xyz` functions.

```
135 pub fn fetch_all_configs(path: String, encryption_key: &String) -> Vec<Config> {
136     let all_configs_raw = fetch_file(path.clone(), encryption_key);
137
138     let individual_configs = all_configs_raw
139         .split("\n")
140         .filter(|line| line.chars().count() >= 3);
141
142     let mut final_configs: Vec<Config> = Vec::<Config>::new();
143
144     for config in individual_configs {
145         let current_config = config.split("|").collect::<Vec<&str>>();
146
147         let tmp_config = Config::create_no_check(current_config[0], current_config[1]);
148         final_configs.push(tmp_config);
149     }
150
151     final_configs
152 }
```

```
154 pub fn save_all_configs(configs: &Vec<Config>, path: String, encryption_key: &String) {
155     let mut stringified_configs = String::new();
156
157     for config in configs {
158         stringified_configs = format!(
159             "{}{}|{}",
160             stringified_configs,
161             if stringified_configs.chars().count() > 1 {
162                 "\n"
163             } else {
164                 ""
165             },
166             config.name,
167             config.value,
168         );
169     }
170
171     save_file(path, stringified_configs, encryption_key);
172     println!("Configs saved!");
173 }
```

Foundations: Redeeming Features

- Unit tests since the beginning for all components.
- Password hashing for users.
- Regexes in use for validating email addresses.
- Initializers for creating sample data on boot.
- Introduction of a CI/CD pipeline to test and deploy after each new commit.
- Encrypting sensitive data such as config values.

```
46     pub fn encrypt(data: String, key: &str) -> String {
47         let mc = new_magic_crypt!(key, 256);
48         let ciphertext = mc.encrypt_str_to_base64(data);
49
50         ciphertext
51     }
52
53     pub fn decrypt(data: String, key: &str) -> Result<EncryptionKey, String> {
54         let mc = new_magic_crypt!(key, 256);
55         let original_data = mc.decrypt_base64_to_string(&data);
56
57         if let Err(e) = original_data {
58             return Err(e.to_string());
59         }
60
61         Ok(EncryptionKey(original_data.unwrap()))
62     }
63 }
```

```
260 pub fn initialize_encryption_key(
261     mappings: &Vec<Mapping>,
262     password: &str,
263 ) -> Result<String, String> {
264     let encryption_key_path = get_file_name("encryption_key", mappings);
265     let encryption_key: Result<String, String>;
266
267     if let Err(e) = encryption_key_path {
268         return Err(e);
269     }
270
271     encryption_key = fetch_encryption_key(encryption_key_path.clone().unwrap(), password);
272
273     if let Err(_) = encryption_key {
274         // Encryption key most likely doesn't exist yet
275         let new_encryption_key = EncryptionKey::generate(20);
276         let saved_encryption_key = save_encryption_key(
277             new_encryption_key.0.clone(),
278             password,
279             &*encryption_key_path.unwrap(),
280         );
281
282         if let Err(f) = saved_encryption_key {
283             return Err(String::from(f));
284         }
285
286         println!("Encryption Key Saved!");
287
288         return Ok(new_encryption_key.0);
289     }
290
291     Ok(encryption_key.unwrap())
292 }
```

Foundations: WASM

The API layer was meant to interact with Kinesis DB using WASM.

All functions and components needed to have separate binding functions.

Problem: It just created yet another unneeded layer and unnecessary complexity. Bridging a JS backend to another Rust backend didn't feel noticeably faster either.

```
20     "dependencies": {
21         "kinesis-db": "file:../pkg"
22     },
```

package.json

```
1  import { fetch_users } from '../pkg/kinesis_db.js';
2
3  let users: any = fetch_users(
4      '/home/edgeking810/Documents/Rust/kinesis-db/data/users.txt',
5      'OhgENBmdz38J0CsRYehi53YCLQFbj99x6ua/dxI5kc8='
6  );
7  console.log(users);
```

```
45 #[wasm_bindgen]
46 pub fn create_config(
47     configs: &str,
48     name: &str,
49     value: &str,
50 ) -> String {
51     let mut all_configs = get_configs_from_str(configs);
52     let result = Config::create(&mut all_configs, name, value);
53
54     if let Err(e) = result {
55         return e;
56     }
57
58     String::from("OK")
59 }
60
61 #[wasm_bindgen]
62 pub fn update_config_value(
63     configs: &str,
64     name: &str,
65     value: &str,
66 ) -> String {
67     let mut all_configs = get_configs_from_str(configs);
68     let result = Config::update_value(&mut all_configs, name, value);
69
70     if let Err(e) = result {
71         return e;
72     }
73
74     String::from("OK")
75 }
76
77 #[wasm_bindgen]
78 pub fn delete_config(
79     configs: &str,
80     name: &str,
81 ) -> String {
82     let mut all_configs = get_configs_from_str(configs);
83     let result = Config::delete(&mut all_configs, name);
84
85     if let Err(e) = result {
86         return e;
87     }
88
89     String::from("OK")
90 }
```



API & LOGIC EXPLOSION

2022

API & Logic Explosion: Rocket

- Why not build the API layer in Rust itself?
- Added serde for the [de]serialization of components.
- Introduced middlewares for creating and verifying JWTs.

```
49 pub fn create_jwt(mappings: &Vec<Mapping>, uid: String) -> Result<String, String> {
50     let expire = match get_config_value(mappings, "JWT_EXPIRE", "60").parse::<i64>() {
51         Ok(val) => val,
52         _ => 60,
53     };
54     let secret = get_config_value(mappings, "TOKEN_KEY", "secret");
55
56     let expiration = Utc::now()
57         .checked_add_signed(chrono::Duration::seconds(expire))
58         .expect("Valid Timestamp")
59         .timestamp();
60
61     let claims = Claims {
62         uid: uid.to_owned(),
63         exp: expiration as usize,
64     };
65
66     let header = Header::new(Algorithm::HS512);
67
68     return match encode(
69         &header,
70         &claims,
71         &EncodingKey::from_secret(secret.as_bytes()),
72     ) {
73         Ok(jwt) => Ok(jwt),
74         Err(_) => Err(String::from("Error: Failed to create JWT")),
75     };
76 }
```

```
78 pub async fn verify_jwt(uid: String, token: String) -> Result<String, (usize, String)> {
79     let mappings = auto_fetch_all_mappings();
80     let secret = get_config_value(&mappings, "TOKEN_KEY", "secret");
81     let users = match auto_fetch_all_users(&mappings) {
82         Ok(u) => u,
83         _ => {
84             return Err((500, String::from("Error: Failed fetching users")));
85         }
86     };
87
88     if !User::exist(&users, &uid) {
89         return Err((404, String::from("Error: Account with this uid not found")));
90     }
91
92     let mut proper_token = "";
93     for v in token.split(" ") {
94         proper_token = v;
95     }
96
97     let decoded = match decode::<Claims>(
98         &proper_token,
99         &DecodingKey::from_secret(secret.as_bytes()),
100         &Validation::new(Algorithm::HS512),
101     ) {
102         Ok(dec) => dec,
103         Err(e) => {
104             return Err((
105                 500,
106                 format!("{}", e), String::from("Error: Failed decoding JWT"), e,
107             ));
108         }
109     };
110
111     if decoded.claims.uid != uid {
112         return Err((403, String::from("Error: Incorrect UID")));
113     }
114
115     Ok(String::from("Successfully Authenticated!"))
116 }
```

API & Logic Explosion: API Routes

- Added routes for fetching, creating, updating and deleting core components.
- Make use of the lettre crate for sending emails (e.g. during user registration process).
- Custom pagination function for fetch routes with offset and limit.

```
55     let all_configs = match auto_fetch_all_configs(&mappings) {  
56         Ok(u) => u,  
57         _ => {  
58             return json!({"status": 500, "message": "Error: Failed fetching configs"});  
59         }  
60     };  
61     let amount = all_configs.len();  
62     let processed_configs = paginate(all_configs, passed_limit, passed_offset);  
63  
64     return json!({"status": 200, "message": "Configs successfully fetched!", "configs": processed_configs, "amount": amount});  
65 }
```

API & Logic Explosion: New Components

New components were introduced along with tests, routes and utils.

```
7  #[derive(Default, Debug, Clone, Serialize, Deserialize)]
8  pub struct Constraint {
9      pub component_name: String,
10     pub properties: Vec<ConstraintProperty>,
11 }
12
```

```
4  #[derive(Default, Debug, Clone, Serialize, Deserialize)]
5  pub struct ConstraintProperty {
6      pub property_name: String,
7      pub is_alphabetic: bool,
8      pub is_numeric: bool,
9      pub min: usize,
10     pub max: usize,
11     pub not_allowed: Vec<char>,
12     pub additional_allowed: Vec<char>,
13 }
```

```
13 #[derive(Default, Debug, Clone, Deserialize, Serialize, PartialEq)]
14 pub struct Event {
15     pub id: String,
16     pub event_type: String,
17     pub description: String,
18     pub timestamp: String,
19     pub redirect: String,
20 }
```

```
10 #[derive(Default, Debug, Clone, Serialize, Deserialize)]
11 pub struct Data {
12     pub id: String,
13     pub project_id: String,
14     pub collection_id: String,
15     pub pairs: Vec<DataPair>,
16     pub published: bool,
17 }
```

```
7  #[derive(Default, Debug, Clone, Serialize, Deserialize)]
8  pub struct DataPair {
9      pub id: String,
10     pub structure_id: String,
11     pub custom_structure_id: String,
12     pub value: String,
13     pub dtype: String,
14 }
15
```

```
9  #[derive(Default, Debug, Clone, Serialize, Deserialize)]
10 pub struct Media {
11     pub id: String,
12     name: String,
13 }
```

```
3  #[derive(Default, Debug, Clone, Serialize, Deserialize, PartialEq)]
4  pub struct RawPair {
5      pub data_id: String,
6      pub structures: Vec<StructurePair>,
7      pub custom_structures: Vec<CustomStructurePair>,
8      pub published: bool,
9  }
10
11 #[derive(Default, Debug, Clone, Serialize, Deserialize, PartialEq)]
12 pub struct StructurePair {
13     pub id: String,
14     pub value: String,
15     pub rtype: String,
16 }
17
18 #[derive(Default, Debug, Clone, Serialize, Deserialize, PartialEq)]
19 pub struct CustomStructurePair {
20     pub id: String,
21     pub structures: Vec<StructurePair>,
22 }
```

API & Logic Explosion: Notable Changes

- Remove wasm since it was too buggy, had an overhead and KDB was behaving like an API for another Backend.
- Codebase restructuring: splitting files into smaller specialized files and group similar files in 'categories' (e.g. components/).
- Introduction of utils: utility functions to simplify recurring operations (such as fetching all 'records' of a component).
- Addition of the 'VIEWER' role for users.
- Custom Dockerfile and enhanced CI/CD for packaging, pushing and deploying to a K8s cluster.
- .lock files for reading/writing from/to disk (I/O) and several utility functions (e.g. creating nested directories).
- New fields for existing core components.
- Caching data using Redis as a POC.
- KDL as a language to represent routing objects.
- REPL as a POC for selecting data.

```
1 use rocket::fairing::{Fairing, Info, Kind};
2 use rocket::http::Header;
3 use rocket::{Request, Response};
4
5 pub struct CORS;
6
7 #[rocket::async_trait]
8 impl Fairing for CORS {
9     fn info(&self) -> Info {
10         Info {
11             name: "Attaching CORS headers to responses",
12             kind: Kind::Response,
13         }
14     }
15
16     async fn on_response<'r>(&self, _request: &'r Request<'_, response: &mut Response<'r>) {
17         response.set_header(Header::new("Access-Control-Allow-Origin", "*"));
18         response.set_header(Header::new(
19             "Access-Control-Allow-Methods",
20             "GET, POST, OPTIONS",
21         ));
22         response.set_header(Header::new("Access-Control-Allow-Headers", "*"));
23         response.set_header(Header::new("Access-Control-Allow-Credentials", "true"));
24     }
25 }
```

Added a CORS Middleware too although it wasn't amazing.

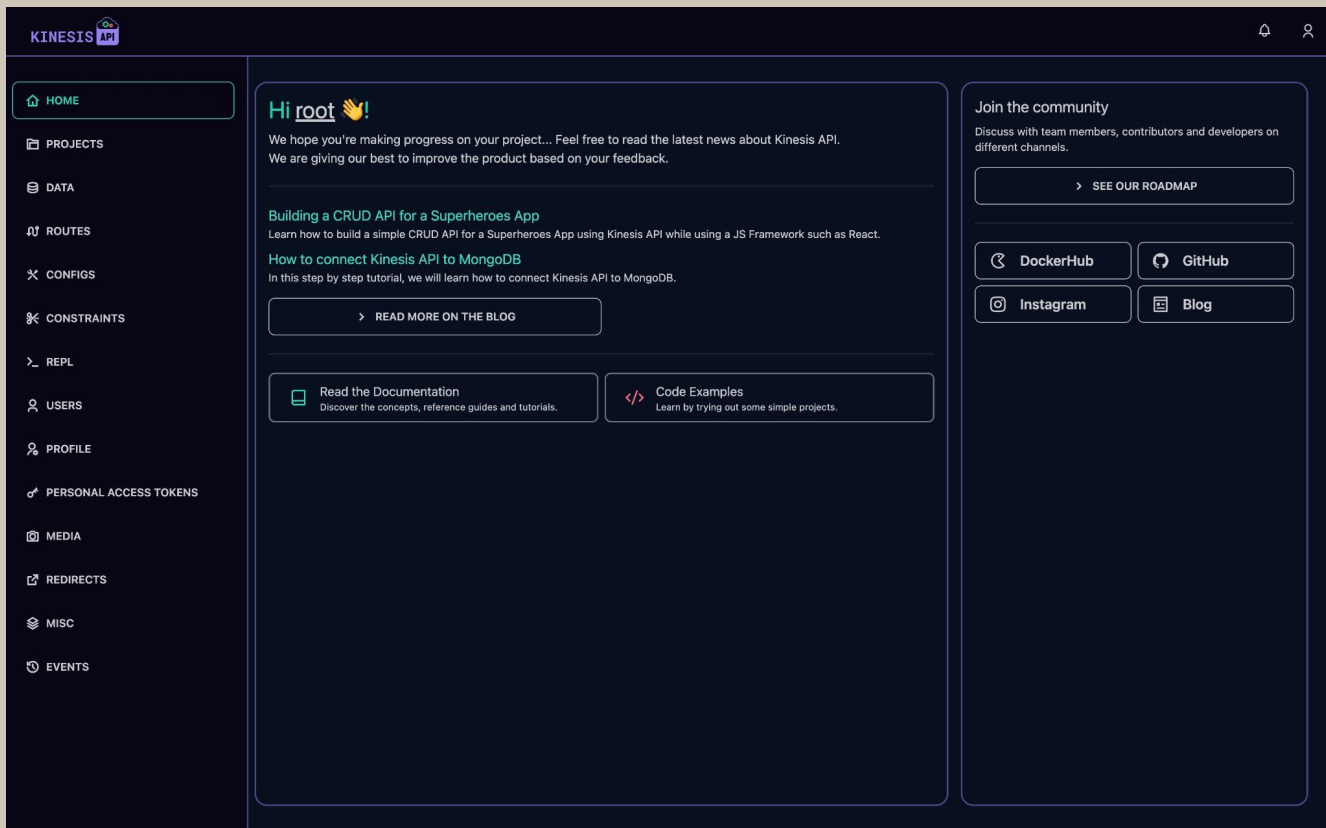
API & Logic Explosion: Complications

- Complicated logic in I/O to handle file and directory operations.
- Make sure data pairs respect constraints of structures when they're created.
- Logic to delete linked components (e.g. Project → Collections + Structures + Custom Structures).
- Similar code for updating all components when a linked field changes (e.g. collection_id).

```
281 -     if !String::from(description)
282 -         .chars()
283 -         .all(|c| c != ';' && c != '>' && c != '#')
284 -     {
285 -         return Err((
286 -             400,
287 -             String::from("Error: description contains an invalid character"),
288 -         ));
289 -     }
290 -
291 -     if String::from(description.trim()).len() < 1 {
292 -         return Err((
293 -             400,
294 -             String::from("Error: description does not contain enough characters"),
295 -         ));
296 -     } else if String::from(description.trim()).len() > 400 {
297 -         return Err((
298 -             400,
299 -             String::from("Error: description contains too many characters"),
300 -         ));
301 -     }
302 +
303 +     let mappings = auto_fetch_all_mappings();
304 +     let all_constraints = match auto_fetch_all_constraints(&mappings) {
305 +         Ok(c) => c,
306 +         Err(e) => return Err((500, e)),
307 +     };
308 +
309 +     let final_value = match ConstraintProperty::validate(&all_constraints, "collection", "description", description) {
310 +         Ok(v) => v,
311 +         Err(e) => return Err(e),
312 +     };
313 +
314 +     let mut all_collections = match auto_fetch_all_collections(&mappings) {
315 +         Ok(u) => u,
316 +         _ => {
317 +             return json!({"status": 500, "message": "Error: Failed fetching collections"});
318 +         }
319 +     };
320 +
321 +     Collection::delete_by_project(&mut all_collections, &passed_project_id);
322 +
323 +     match auto_save_all_collections(&mappings, &all_collections) {
324 +         Err(e) => return json!({"status": 500, "message": e}),
325 +         _ => {}
326 +     }
327 +
328 +     if let Err(e) = auto_create_event(
329 +         &mappings,
330 +         "project_delete",
331 +         format!(
332 +             "The project <{}> was deleted by user[{}]",
333 +             project.name, &passed_uid
334 +         ),
335 +         format!("/projects"),
336 +     ) {
337 +         return json!({"status": e.0, "message": e.1});
338 +     }
339 +
340 +     match auto_save_all_projects(&mappings, &all_projects) {
341 +         Ok(_) => return json!({"status": 200, "message": "Project successfully deleted!")),
342 +         Err(e) => {
343 +             json!({"status": 500, "message": e})
344 +         }
345 +     }
346 + }
```

```
78 match Project::delete(&mut all_projects, &passed_project_id) {
79     Err(e) => return json!({"status": e.0, "message": e.1}),
80     _ => {}
81 }
82
83 let mut all_collections = match auto_fetch_all_collections(&mappings) {
84     Ok(u) => u,
85     _ => {
86         return json!({"status": 500, "message": "Error: Failed fetching collections"});
87     }
88 };
89
90 Collection::delete_by_project(&mut all_collections, &passed_project_id);
91
92 match auto_save_all_collections(&mappings, &all_collections) {
93     Err(e) => return json!({"status": 500, "message": e}),
94     _ => {}
95 }
96
97 if let Err(e) = auto_create_event(
98     &mappings,
99     "project_delete",
100     format!(
101         "The project <{}> was deleted by user[{}]",
102         project.name, &passed_uid
103     ),
104     format!("/projects"),
105 ) {
106     return json!({"status": e.0, "message": e.1});
107 }
108
109 match auto_save_all_projects(&mappings, &all_projects) {
110     Ok(_) => return json!({"status": 200, "message": "Project successfully deleted!")),
111     Err(e) => {
112         json!({"status": 500, "message": e})
113     }
114 }
115 }
```

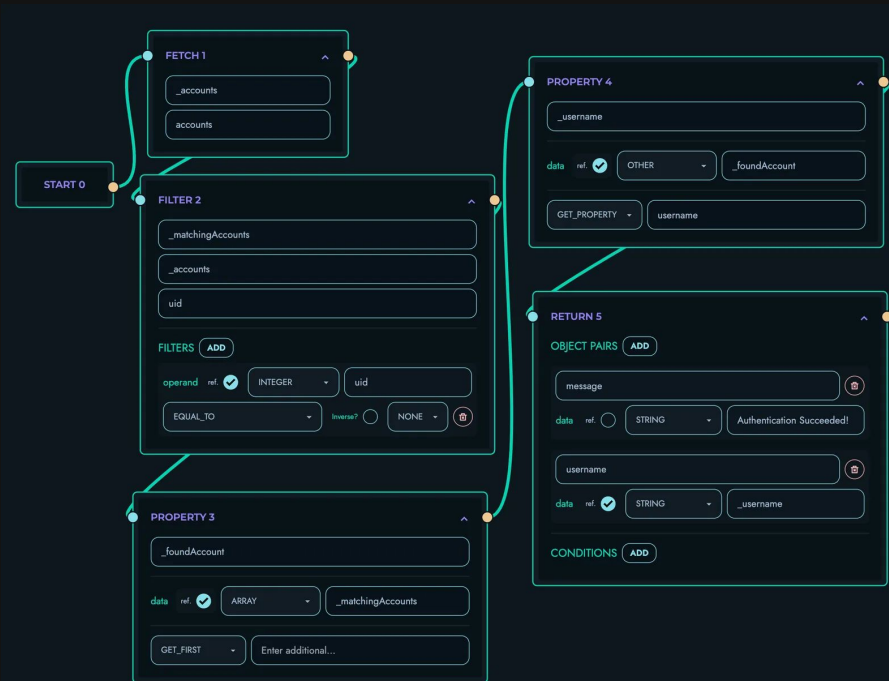
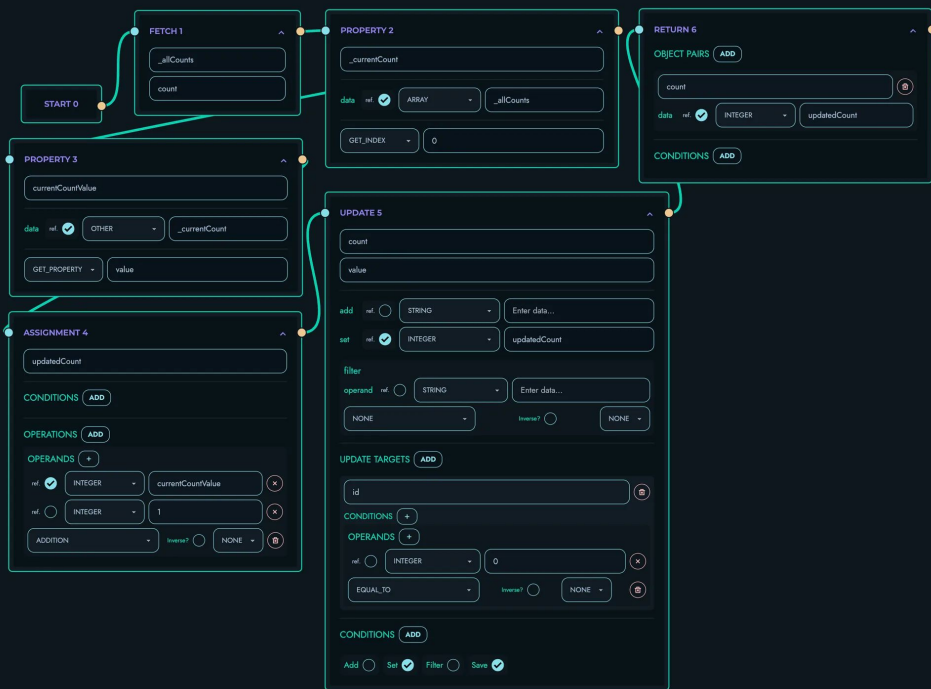
Kinesis API Portal



- ReactJS
- Ugly af
- Scaling limitations
- Different codebase
- Replaced by Tera

The X Engine

The magic behind "the API to make an API".



X Engine : Components

```
11 #[derive(Default, Debug, Clone, Serialize, Deserialize, PartialEq)]
12 pub struct RouteComponent {
13     pub route_id: String,
14     pub route_path: String,
15     pub project_id: String,
16     pub auth_jwt: Option<AuthJWT>,
17     pub body: Vec<BodyData>,
18     pub params: Option<ParamData>,
19     pub flow: RouteFlow,
20 }
```

```
3 #[derive(Default, Debug, Clone, Serialize, Deserialize, PartialEq)]
4 pub struct AuthJWT {
5     pub active: bool,
6     pub field: String,
7     pub ref_col: String,
8 }
```

```
4 #[derive(Default, Debug, Clone, Serialize, Deserialize, PartialEq)]
5 pub struct BodyData {
6     pub id: String,
7     pub bdtype: BodyDataType,
8 }
```

```
5 #[derive(Default, Debug, Clone, Serialize, Deserialize, PartialEq)]
6 pub struct ParamData {
7     pub delimiter: String,
8     pub pairs: Vec<BodyData>,
9 }
```

```
10 #[derive(Default, Debug, Clone, Serialize, Deserialize, PartialEq)]
11 pub struct RouteFlow {
12     pub fetchers: Vec<FetchBlock>,
13     pub assignments: Vec<AssignmentBlock>,
14     pub templates: Vec<TemplateBlock>,
15     pub conditions: Vec<ConditionBlock>,
16     pub loops: Vec<LoopBlock>,
17     pub filters: Vec<FilterBlock>,
18     pub properties: Vec<PropertyBlock>,
19     pub functions: Vec<FunctionBlock>,
20     pub objects: Vec<ObjectBlock>,
21     pub updates: Vec<UpdateBlock>,
22     pub creates: Vec<CreateBlock>,
23     pub returns: Vec<ReturnBlock>,
24 }
```

Several modules and submodules required for representing and storing routes created by users.

body_data_type.rs

condition_action.rs

condition_plain.rs

condition_type.rs

condition.rs

fail_obj.rs

filter.rs

function_list.rs

function.rs

next_condition_type.rs

object_pair.rs

operation_type.rs

operation.rs

property_apply.rs

property.rs

ref_data.rs

submodules.rs

update_target.rs

X Engine : The /x route

- A generic, configurable API gateway used across projects.
- Supports GET, POST, PUT, PATCH, DELETE, HEAD and OPTIONS verbs.
- Centralized 'handle_request' method for orchestrating processing.
- Finds out which project's API path and route is after the "/x/".
- Fetches and caches all relevant data pairs for the target project.
- Validates URL parameters and body parameters that have been passed.
- Checks the JWT authentication mechanism, if enforced.
- GlobalBlockOrder: Obtains an ordered list of blocks for the current route.
- LoopProcessor: Checks for loops and processes them.
- DefinitionStore: Adds definitions for all blocks during block processing.
- Resolvers: Resolve conditions, operations and referenced data during processing.
- SignalProcessor: Returns and parses signals after processing blocks.

```
16 pub struct GlobalBlockOrder {
17     pub index: usize,
18     pub block_index: usize,
19     pub global_index: usize,
20     pub name: String,
21     pub ref_name: String,
22 }
```

```
25 pub struct LoopObject {
26     pub start_index: usize,
27     pub end_index: usize,
28     pub loop_index: usize,
29     pub ref_var: String,
30     pub parent: Option<String>,
31 }
```

```
16 pub enum Signal {
17     NONE,
18     BREAK,
19     CONTINUE,
20     FAIL(usize, String),
21     RETURN(Value),
22 }
```

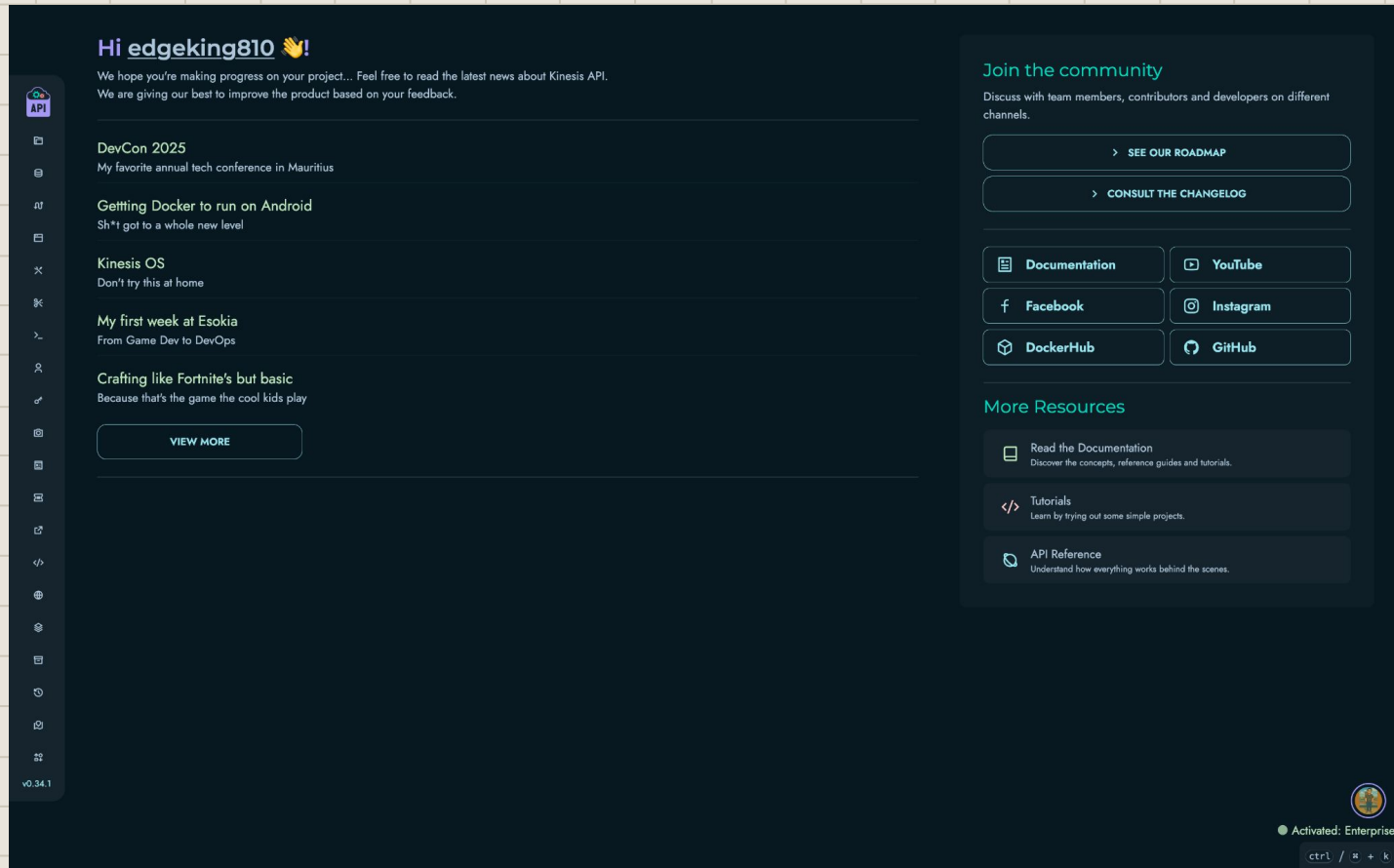
```
21 #[derive(Debug, Clone, Serialize, Deserialize)]
22 pub struct DefinitionStore {
23     pub name: String,
24     pub global_index: usize,
25     pub block_index: usize,
26     pub data: DefinitionData,
27 }
28
29 #[derive(Debug, Clone, Serialize, Deserialize, PartialEq)]
30 pub enum DefinitionData {
31     NULL,
32     UNDEFINED,
33     BOOLEAN(bool),
34     STRING(String),
35     INTEGER(usize),
36     FLOAT(f64),
37     ARRAY(Vec<DefinitionData>),
38     DATA(RawPair),
39     NoChange,
40 }
```



STABILIZATION & UX

2024

Stabilization & UX: Web UI



New Web UI built using
Tera templates along
with DaisyUI.

All components could
now be managed from
the Web UI itself.

Stabilization & UX: I/O Performance

```
154 pub fn construct_block(properties: HashMap<String, String>) -> String {
155     let mut block = String::new();
156
157     for (k, v) in properties {
158         let new_value = v.replace("\n", "0x0a").replace("=====", "");
159         block.push_str(&format!("{}", k, new_value));
160     }
161
162     block
163 }
164
165 pub fn construct_blocks(blocks: Vec<String>, property: Option<String>) -> String {
166     let mut final_block = String::new();
167     let _property = match property {
168         Some(p) => p.clone(),
169         None => String::new(),
170     };
171
172     if _property.len() > 0 {
173         final_block.push_str(&format!("<{}>\n", _property));
174     }
175
176     for (i, b) in blocks.iter().enumerate() {
177         final_block.push_str(&format!("{}", b));
178         if i < blocks.len() - 1 {
179             if _property.len() > 0 {
180                 final_block.push_str(&format!("_{}_{}\n", _property));
181             } else {
182                 final_block.push_str("$_=====$\n");
183             }
184         }
185     }
186
187     if _property.len() > 0 {
188         final_block.push_str(&format!("{}", _property));
189     }
190
191     final_block
192 }
193
194 // <property=
195 // __property__
196 // property>
```

Replace Redis POC by a custom built
Memory DB: Kasper V2 Engine.

Add I/O BlockEngine for saving
components in a more generic format.

```
111 pub fn get_memory_db(key: String) -> Option<String> {
112     unsafe {
113         match memory_db(key, None) {
114             Some(v) => Some(v),
115             None => None,
116         }
117     }
118 }
119
120 pub fn set_memory_db(key: String, value: String) -> String {
121     unsafe {
122         match memory_db(key, Some(value)) {
123             Some(v) => v,
124             None => String::new(),
125         }
126     }
127 }
```

```
21 pub unsafe fn memory_db(key: String, value: Option<String>) -> Option<String> {
22     static mut MEMORY_DB: OnceLock<Mutex<HashMap<String, String>>> = OnceLock::new();
23
24     let handle =
25         thread::spawn(
26             move || match MEMORY_DB.get_or_init(|| Mutex::new(HashMap::new())) { lock() {
27                 Ok(mut guard) => {
28                     let mem_db = guard.deref_mut();
29
30                     match value {
31                         Some(v) => {
32                             mem_db.insert(key, v);
33                             mem_db.insert("_return".to_string(), String::from("OK!"));
34                         }
35                         None => match mem_db.get(&key) {
36                             Some(v) => {
37                                 mem_db.insert("_return".to_string(), v.clone());
38                             }
39                             None => {
40                                 mem_db.insert("_return".to_string(), String::new());
41                             }
42                         },
43                     }
44                 },
45                 _ => println!("Failed acquiring memory_db lock."),
46             },
47         );
48
49     match handle.join() {
50         Ok(_) => match MEMORY_DB
51             .get()
52             .unwrap()
53             .lock()
54             .unwrap()
55             .borrow()
56             .get("_return")
57         {
58             Some(v) => {
59                 if v == "" {
60                     return None;
61                 }
62                 return Some(v.clone());
63             }
64             None => None,
65         },
66         Err(_) => {
67             println!("Failed to join memory_db thread.");
68             return None;
69         }
70     }
71 }
```

Stabilization & UX: New Components

New components: UUID, PersonalAccessToken, Redirect & Snippet

```
16  #[derive(Default, Debug, Clone, Serialize, Deserialize, PartialEq, ToSchema)]
17  pub struct PersonalAccessToken {
18      /// A unique auto-generated identifier.
19      #[schema(example = "39b6f437-7f7b-44d5-8960-5e76be09400e")]
20      pub id: String,
21      /// An associated name or message with the personal access token.
22      #[schema(example = "test pat")]
23      name: String,
24      /// The owner of the personal access token (user.id).
25      #[schema(example = "15a70484-6684-4eb0-a6d9-6266b2719261")]
26      pub owner_id: String,
27      /// The timestamp that defines since when the personal access token is valid.
28      #[schema(example = "2024-07-21T18:43:00+04:00")]
29      valid_from: String,
30      /// The timestamp that defines until when the personal access token is valid.
31      #[schema(example = "2024-07-21T18:44:00+04:00")]
32      pub valid_to: String,
33      /// The personal access token itself (hashed).
34      #[schema(
35          example = "$argon2id$v=19$m=19456,t=2,p=1$yvnQn/ekFRo0iom/t1qUEYg$0f8Iy2qagcMM6cxQRbUiSSKBiLZ/+tPbJhU3IrLKaNy"
36      )]
37      pub token: String,
38      /// The rights or permissions associated with the personal access token.
39      pub rights: Vec<TokenRight>,
40  }
41
42  #[allow(non_camel_case_types)]
43  #[derive(Debug, Clone, Serialize, Deserialize, PartialEq, ToSchema)]
44  pub enum TokenRight {
45      USER_FETCH,          // ROOT or owner_id == update_id
46      USER_CREATE,         // ROOT
47      USER_UPDATE,         // owner_id == update_id
48      USER_UPDATE_ROLE,    // ROOT
49      USER_DELETE,         // ROOT
50      CONSTRAINT_FETCH,    // Any
51      CONSTRAINT_UPDATE,   // ROOT
52      CONFIG_FETCH,        // ROOT
53      CONFIG_CREATE,       // ROOT
54      CONFIG_UPDATE,       // ROOT
55      CONFIG_DELETE,       // ROOT
```

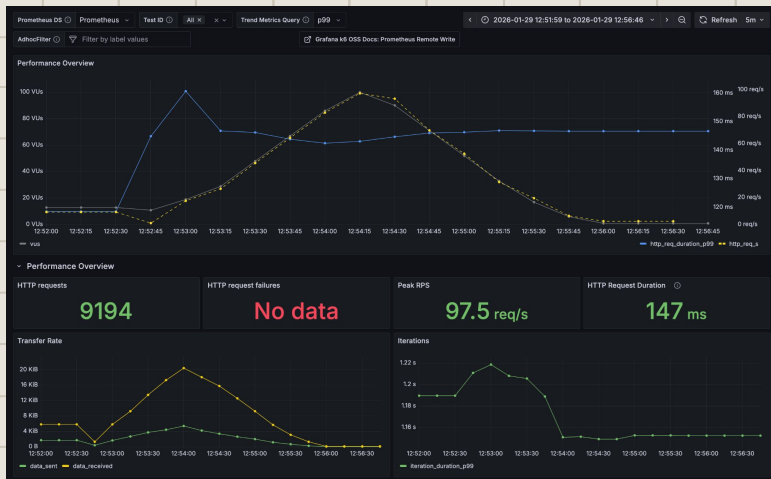
```
13  pub struct Redirect {
14      /// The id of the redirect.
15      #[schema(example = "abc")]
16      pub id: String,
17      /// The url of the redirect.
18      #[schema(example = "https://api.kinesis.world")]
19      pub url: String,
20  }
```

```
13  pub struct Snippet {
14      /// The id of the snippet.
15      #[schema(example = "abc")]
16      pub id: String,
17      /// The uid of the user who created the snippet.
18      #[schema(example = "15a70484-6684-4eb0-a6d9-6266b2719261")]
19      pub uid: String,
20      /// The name of the snippet.
21      #[schema(example = "test_snippet")]
22      pub name: String,
23      /// A description for the snippet.
24      #[schema(example = "Just a snippet for testing purposes.")]
25      pub description: String,
26      /// Whether the snippet should be publicly visible or not.
27      #[schema(example = false)]
28      pub is_public: bool,
29      /// A timestamp that defines the lifetime of the snippet.
30      #[schema(example = "2024-07-21T18:44:00+04:00")]
31      pub expiry: Option<String>,
32      /// The actual content of the snippet.
33      #[schema(example = "# Hello World!")]
34      pub content: String,
35  }
```

Stabilization & UX: General Improvements

- Move to gitea instead of gitlab.
- Improve docker build and release process.
- X Engine refactoring.
- Utilization of cookies for the /web endpoint.
- Removal of the REPL feature.
- Improve SMTP logic.
- Add /metrics route for prometheus.
- Initialize the roadmap page on the Web UI.
- Introduction of load testing using k6.
- Add pagination for components on the Web UI.
- New route for resetting user passwords.

- New middleware for validating rights through PAT and JWT.
- API Documentation using utoipa and Scalar.
- Reduce calls to auto_fetch_xxx functions.
- Use create_no_check functions when reading existing data.
- Bug fixes.



The image shows the Scalar API documentation for the 'Forget Password' endpoint. The endpoint is a POST request to `/user/forget/password`. The body is a JSON object with the following fields:

- `auth_data` (string, required): Username or email address of the user.

The response is a 200 status code with a JSON object containing the following fields:

- `message` (string): "Email with a link to reset the password sent!"
- `status` (integer): 200

The documentation also includes a 'Test Request' button and a 'Show Schema' button.

Stabilization & UX : X Engine

- Add support for resolving multiple operands in conditions and operations.
- Redesign functionality of the DefinitionStore.
- Updated block: LoopBlock
- New block: EndLoopBlock

```
14 pub struct LoopBlock {
15     pub global_index: u32,
16     pub block_index: u32,
17     pub local_name: String,
18     pub start: RefData,
19     pub end: RefData,
20     pub step: Option<RefData>,
21     pub include_last: bool,
22 }
```

```
11 pub struct EndLoopBlock {
12     pub global_index: u32,
13     pub block_index: u32,
14     pub local_name: String,
15 }
```

```
19 pub fn detect_loops(
20     global_blocks: &Vec<GlobalBlockOrder>,
21     current_block: &GlobalBlockOrder,
22 ) -> Result<Vec<LoopObject>, (usize, String)> {
23     let mut loop_objects = Vec::<LoopObject>::new();
24
25     let block_index = current_block.block_index;
26     let mut start_index = 0;
27     let mut is_processing_block = false;
28     let mut loop_var = String::new();
29
30     // Loop through all the blocks in global blocks
31     for (i, block) in global_blocks.iter().enumerate() {
32         // If a loop is being processed, and another loop is found or another block_index has
33         // been reached, the current loop is completed.
34         if is_processing_block && (block.name == "LOOP" || block.block_index > block_index) {
35             loop_objects.push(LoopObject {
36                 start_index: start_index,
37                 end_index: i,
38                 ref_var: loop_var.clone(),
39             });
40             is_processing_block = false;
41             start_index = 0;
42         }
43
44         // If a loop is found, set is_processing to true in order to process it and whatever
45         // other blocks are found in it.
46         if block.block_index == block_index && block.name == "LOOP" {
47             is_processing_block = true;
48             start_index = i;
49             loop_var = block.ref_name.clone();
50         }
51     }
52
53     // If is_processing is true, it means that the last loop was not closed yet, and since
54     // the last index of global_blocks has been reached, it means that the last loop should
55     // end at the end of the global_blocks.
56     if is_processing_block {
57         loop_objects.push(LoopObject {
58             start_index: start_index,
59             end_index: global_blocks.len(),
60             ref_var: loop_var,
61         });
62     }
63
64     Ok(loop_objects)
65 }
```



PRODUCTIZATION & SCALE

2025

Productization & Scale : Kinesis DB

Core Capabilities

- Multiple Storage Engines: In-Memory (volatile/fast), On-Disk (persistent/durable), and Hybrid (balanced performance with persistence).
- Full ACID Compliance: Atomicity, Consistency, Isolation (4 levels), and Durability with Write-Ahead Logging (WAL).
- Dynamic Schema Management: Runtime schema creation/modification with strong typing, constraints (required, unique, patterns, min/max), and versioning.
- Robust Transactions: Multi-level isolation control, deadlock detection/prevention, and automatic crash recovery.
- Performance Optimizations: LRU buffer pool, configurable caching, automatic **blob storage** for large data, and efficient indexing strategies.
- Interactive REPL Shell: SQL-inspired query syntax with multiple output formats (standard, JSON, table).

Integration & Modernization

- Seamless Kinesis API Integration: Replace legacy .txt file mapping with unified database backend across all components.
- Eliminate Technical Debt: Remove ad-hoc file handling logic and consolidate data management into a single, reliable system.
- BlobStore for Large Data: Efficient storage and retrieval of large binary/string objects with reference counting and garbage collection.
- Configurable Startup: Full environment variable control for storage engine selection, isolation levels, buffer pool sizing, and recovery policies.

Enhanced Query & Search

- Improved REPL Search Capabilities: Add -case-insensitive flag for flexible pattern matching across records.
- Advanced Query Support: Equality, range, pattern matching, and full-text search capabilities.
- Multiple Output Formats: Support JSON, table, and standard output for different use cases.

Productization & Scale : Kinesis DB

Infrastructure Improvements

- Overflow Pages for Large Data: Efficient handling of records spanning multiple pages with linked page chains.
- Memory-Mapped I/O (mmap2): Leverage OS-level memory mapping for faster, low-latency disk operations.
- Page Format Optimization: Structured 16KB page layout with checksums, overflow chain support, and atomic writes.
- Automatic Compaction: Reclaim fragmented space and optimize disk layout over time.
- Configurable Tuning: Buffer pool size, WAL rotation thresholds, and blob storage thresholds all adjustable.

Performance & Reliability

- 4X+ Performance Boost: Load testing confirms significant throughput improvements over legacy systems.
- Zero Data Loss Guarantees: WAL ensures committed data survives crashes; automatic recovery on restart.
- Concurrent Transaction Support: Record-level locking with deadlock detection enables parallel operations.
- Scalable Architecture: Modular design supports future enhancements (distributed transactions, advanced indexing, encryption).
- Comprehensive Testing: Unit, integration, and performance test coverage for all major features.

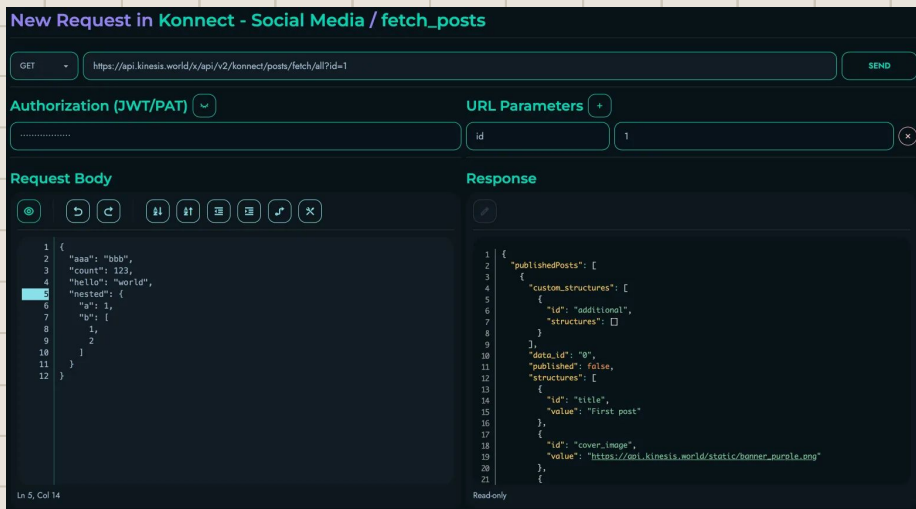
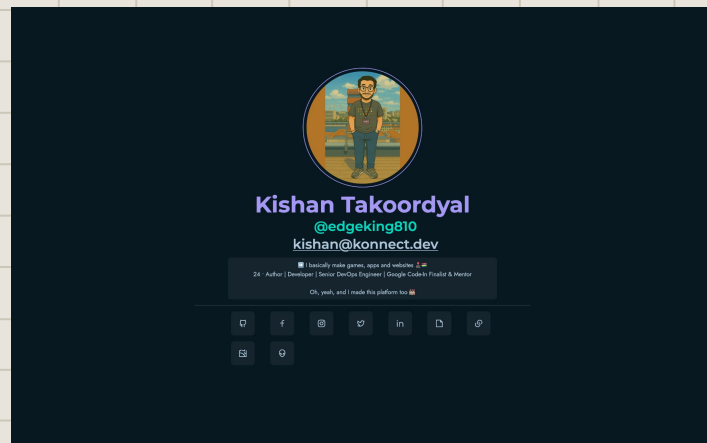
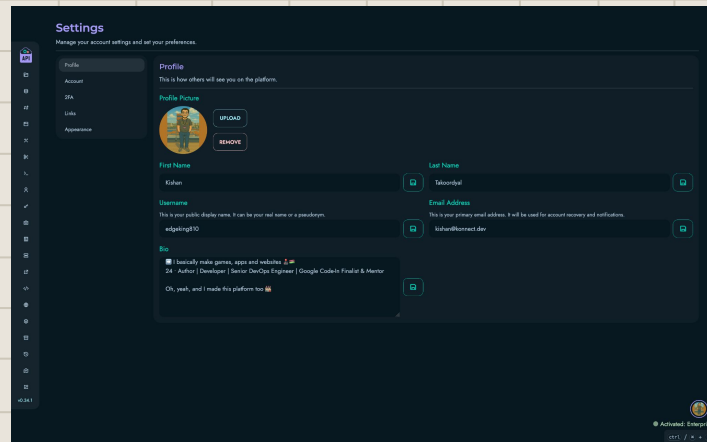
Developer Experience

- Simple Configuration: Single environment variable interface for all tuning needs.
- Rich Diagnostics: Logging, checksums, and consistency checks for debugging.
- Extensibility: Support for custom field types, storage backends, and REPL commands.

*"True reliability isn't built on hope—
it's built on transactions, locks, and
write-ahead logging."*

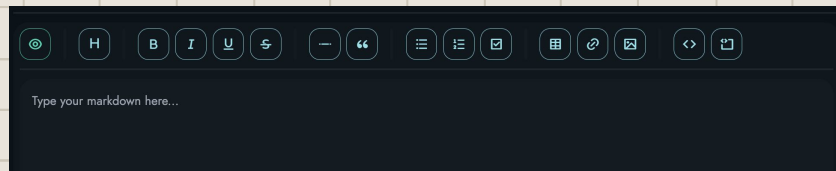
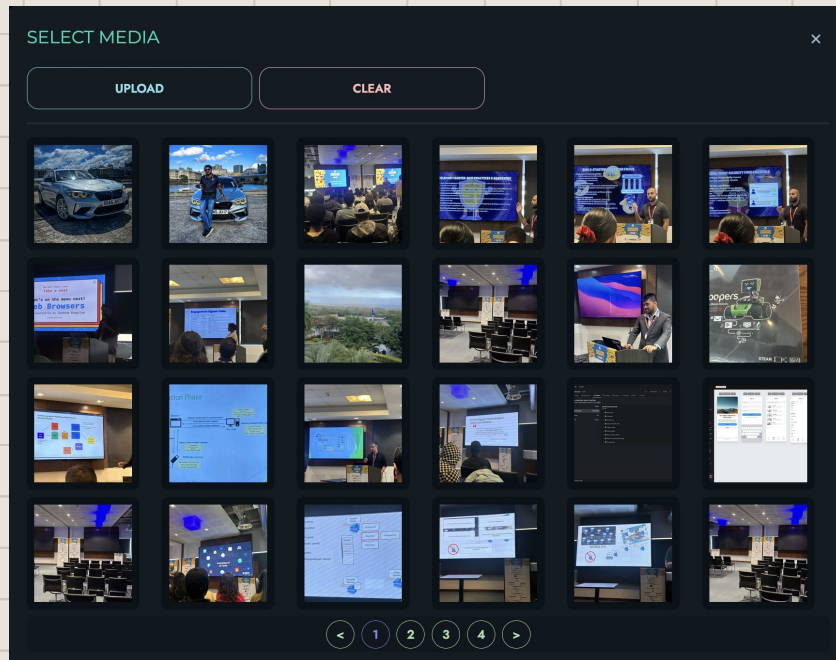
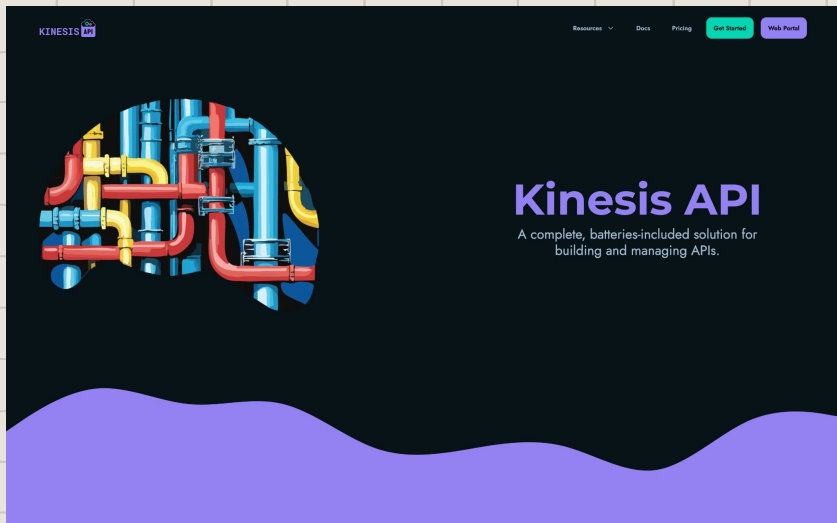
Productization & Scale: Web-focused changes

- The playground page for testing API routes.
- Addition of the `/setup` page for first boot of K. API.
- New `/about` and `/changelog` pages to document releases.
- Implementation of the `/settings` and `/user` pages to allow user customization through bio, social links and profile picture.
- Launch an optional 2FA feature for users.
- Improvements to responsiveness of the Web UI.
- Introduction of the Global Navigate modal.



Productization & Scale: Web-focused changes

- Remake the entire X Engine Routing System UI.
- Phase out the ToastUI Markdown Editor in favor of Markdown IT and other plugins.
- Replace jsoneditor with a custom-made JSON editor.
- SEO improvements & new compression algorithms.
- Addition of the Media Library Popup.
- Introduction of the landing pages including the features, contact, team, pricing, privacy policy, system status, and terms and conditions pages.



Productization & Scale: Documentation

Complete documentation using mdbook for all components including tutorials.

Introduction

- What is Kinesis API?
- Key Features
- Getting Started
- Core Components
- Usage Guides
- Support & Community
- Further Steps

Functionality

- 1. Kinesis DB
- 2. X Engine
- 3. API Reference
- 4. Demo

Getting Started

- 5. First Steps
 - 5.1. Installation
 - 5.2. Initialization
 - 5.3. Setup

Licensing

- 6. Overview
 - 6.1. Accessing the License Server
 - 6.2. Purchasing Licenses
 - 6.3. Registering Licenses

Usage

- 7. Core Components
 - 7.1. Configs
 - 7.2. Constraints
 - 7.3. Users
 - 7.3.1. Personal Access Tokens
 - 7.3.2. Profile
 - 7.4. Projects
 - 7.5. Collections
 - 7.6. Structures
 - 7.6.1. Custom Structures
 - 7.7. Data
 - 7.8. Routes
 - 7.8.1. Blocks
 - 7.8.2. Playground

Menu

Search

Kinesis API Docs

Introduction

Welcome to the official documentation for Kinesis API, a powerful all-in-one framework that transforms how you create, manage, and scale APIs.

What is Kinesis API?

Kinesis API is a comprehensive solution for API development that combines:

- A custom-built, high-performance database system (Kinesis DB)
- A visual editor for creating API routes with the X Engine
- Integrated management tools that eliminate the need for multiple external services

Whether you're prototyping a simple API or building complex, interconnected systems, Kinesis API provides the tools to accelerate development without sacrificing quality or control.

Key Features

- All-in-one Platform:** API creation, database management, and execution in a single unified environment
- Visual Route Builder:** Create complex API logic without writing traditional code using our block-based system
- Custom Database:** Built-in ACID-compliant database system with multiple storage engines and strong schema management
- Performance-Focused:** Developed in Rust for maximum efficiency and reliability
- Flexible Deployment:** Deploy anywhere with our Docker images
- Comprehensive Management:** User authentication, role-based access control, and extensive monitoring capabilities

Getting Started

New to Kinesis API? Start here:

- 1. Installation Guide - Set up Kinesis API on your system
- 2. Initialization - Configure your instance for first use
- 3. API Tutorials - Build your first API with Kinesis API

Core Components

Productization & Scale: Ticketing System

A new ticketing component was introduced along with tests, routes, utils, docs and web pages.

```
24 #[derive(Default, Debug, Clone, Serialize, Deserialize, ToSchema)]
25 pub struct Ticket {
26     /// A unique identifier for the ticket.
27     #[schema(example = 1)]
28     pub id: usize,
29     /// The title of the ticket.
30     #[schema(example = "Example Ticket Title")]
31     pub title: String,
32     /// The description of the ticket.
33     #[schema(example = "This is an example description for the ticket.")]
34     pub description: String,
35     /// The type of the ticket.
36     #[schema(example = "BUG")]
37     pub ticket_type: TicketType,
38     /// The status of the ticket.
39     #[schema(example = "OPEN")]
40     pub status: TicketStatus,
41     /// The priority of the ticket.
42     #[schema(example = "LOW")]
43     pub priority: TicketPriority,
44     /// The ID of the user who created the ticket.
45     #[schema(example = 1)]
46     pub user_id: Option<usize>,
47     /// An optional field for an anonymous user.
48     #[schema(example = "")]
49     pub anonymous_user: Option<TicketAnonymousUser>,
50     /// Assignees for the ticket.
51     #[schema(example = "[]")]
52     pub assignees: Vec<usize>,
53     /// The date and time when the ticket was created.
54     #[schema(example = "2025-06-26T20:45:00+04:00")]
55     pub created_at: String,
56     /// The date and time when the ticket was last updated.
57     #[schema(example = "2025-06-26T20:45:00+04:00")]
58     pub updated_at: String,
59     /// Whether the ticket is publicly visible.
60     #[schema(example = true)]
61     pub public: bool,
62     /// Whether the ticket is archived.
63     #[schema(example = false)]
64     pub archived: bool,
65     /// A list of email addresses associated with the ticket for notifications.
66     #[schema(example = "[]")]
67     pub subscribed_emails: Vec<String>,
68     /// The tags and labels associated with the ticket.
69     #[schema(example = "[]")]
70     pub tags: Vec<String>,
71     /// A list of comments associated with the ticket.
72     #[schema(example = "[]")]
73     pub comments: Vec<TicketComment>,
74 }
```

```
6 #[derive(Debug, Clone, Copy, Serialize, Deserialize, PartialEq, ToSchema)]
7 pub enum TicketStatus {
8     OPEN,
9     ACTIVE,
10    RESOLVED,
11    CLOSED,
12    WONTDO,
13 }

8 #[derive(Debug, Clone, Serialize, Deserialize, PartialEq, ToSchema)]
9 pub struct TicketAnonymousUser {
10     /// The name of the anonymous user
11     #[schema(example = "John Doe")]
12     pub name: String,
13     /// The email of the anonymous user
14     #[schema(example = "john_doe@local.host")]
15     pub email: String,
16 }

6 #[derive(Debug, Clone, Copy, Serialize, Deserialize, PartialEq, ToSchema)]
7 pub enum TicketPriority {
8     /// Represents a ticket with low priority.
9     LOW,
10    /// Represents a ticket with medium priority.
11    MEDIUM,
12    /// Represents a ticket with high priority.
13    HIGH,
14    /// Represents a ticket with critical priority.
15    CRITICAL,
16    /// Represents a ticket with urgent priority.
17    URGENT,
18 }

6 #[derive(Debug, Clone, Copy, Serialize, Deserialize, PartialEq, ToSchema)]
7 pub enum TicketType {
8     /// Represents a bug in the system.
9     BUG,
10    /// Represents a feature request.
11    FEATURE,
12    /// Represents an improvement to an existing feature.
13    IMPROVEMENT,
14    /// Represents a question or inquiry.
15    QUESTION,
16    /// Represents a task to be completed.
17    TASK,
18    /// Represents feedback on the system.
19    FEEDBACK,
20 }
```

```
19 #[derive(Default, Debug, Clone, Serialize, Deserialize, ToSchema)]
20 pub struct TicketComment {
21     /// A unique identifier for the comment.
22     #[schema(example = 1)]
23     pub id: usize,
24     /// The ID of the ticket to which this comment belongs.
25     #[schema(example = 1)]
26     pub ticket_id: usize,
27     /// The content of the comment.
28     #[schema(example = "This is a comment on the ticket.")]
29     pub content: String,
30     /// The ID of the user who created the comment.
31     #[schema(example = 1)]
32     pub user_id: Option<usize>,
33     /// An optional field for an anonymous user.
34     #[schema(example = "")]
35     pub anonymous_user: Option<TicketAnonymousUser>,
36     /// The date and time when the comment was created.
37     #[schema(example = "2025-06-26T20:45:00+04:00")]
38     pub created_at: String,
39     /// The date and time when the comment was last updated.
40     #[schema(example = "2025-06-26T20:45:00+04:00")]
41     pub updated_at: String,
42 }
```

Productization & Scale: Ticketing System

#1 Background process for sending emails



HIGH

ACTIVE

TASK

PUBLIC

Description

Behind the scenes, the API sends out mails for many events such as user registration, when a password reset is requested, etc. Now, with the implementation of the ticketing system, sending mails happens even more often, since assignees and users subscribed to a ticket need to receive notifications when the ticket is updated.

The problem

Sending out a mail takes about 5 seconds, and the way this is implemented right now in the API, it all happens synchronously, which means responses from the API take a couple of seconds more to be returned. Further more, if sending out a mail fails, the API will return a 'negative response' even if every other operation in that specific route's logic successfully ran. Sending out mails also depend on having valid SMTP settings configured in the [configs](#), which don't have anything to do with the logic of routes featuring sending mails.

The solution

Maybe a middleware or an asynchronous function needs to be put in place for sending out mails in the background. Logs and errors while sending mails should be logged somewhere else or appear in the Rocket process' STDOUT/STDERR.

```
1 use rocket::tokio;
2 use rocket::tokio::task;
3 use rocket::response::content;
4
5 #[rocket::get("/")]
6 async fn index() -> content::RawHtml {
7     // This is a simple example; in real applications, you might
8     // be reading from a file or database.
9     let message = task::spawn_blocking(|| {
10         // Simulate a blocking operation
11         std::thread::sleep(std::time::Duration::from_secs(1));
12         "Hello from a blocking thread!"
13     }).await.unwrap();
14
15     content::RawHtml(format!("

> {}</p>", message)).as_str()
16 }
17
18 #[rocket::main]
19 async fn main() -> Result<(), rocket::Error> {
20     let _rocket = rocket::build().mount("/", rocket::routes![index]).launch().await?;
21     Ok(())
22 }


```

Resources

- [Scheduled background task using database in Rocket?](#)
- [Rocket and scheduled tasks](#)
- [Background processes #1640](#)

Author: Kishan Takoordyal (kishan@kconnect.dev)

Created on: Thu Jul 03 2025 17:11:03

Last updated: Sun Feb 08 2026 12:31:29

Tags: [email](#) [medium](#) [task](#)

ASSIGNEES

SUBSCRIBERS

UNSUBSCRIBE

EDIT DESCRIPTION

ARCHIVE TICKET

DELETE TICKET

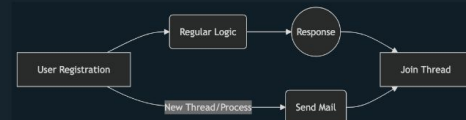
Comments

POST A NEW COMMENT

Kishan Takoordyal (kishan@kconnect.dev)

Posted on: Thu Jul 03 2025 17:12:25 (edited)

Resolved and released in the **v0.25.0** update



Activated: Enterprise

ctrl / ⌘ + k

Productization & Scale: Blog Engine

New blogging components were introduced along with tests, routes, utils, docs and web pages.

```
18 #[derive(Default, Debug, Clone, Serialize, Deserialize, ToSchema)]
19 pub struct BlogPost {
20     /// A unique identifier for the blog post.
21     #[schema(example = 1)]
22     pub id: usize,
23     /// A slug for the blog post, used in URLs.
24     #[schema(example = "example-blog-post")]
25     pub slug: String,
26     /// The title of the blog post.
27     #[schema(example = "Example Blog Post Title")]
28     pub title: String,
29     /// The subtitle of the blog post.
30     #[schema(example = "This is an example subtitle for the blog post.")]
31     pub subtitle: String,
32     /// A preview image URL for the blog post.
33     #[schema(example = "https://example.com/image.jpg")]
34     pub preview_image: String,
35     /// Images forming part of a carousel for the blog post.
36     #[schema(example = "[ ]")]
37     pub carousel_images: Vec<String>,
38     /// The content of the blog post.
39     #[schema(example = "This is the content of the blog post.")]
40     pub content: String,
41     /// The ID of the user who created the blog post.
42     #[schema(example = 1)]
43     pub user_id: usize,
44     /// The date and time when the blog post was created.
45     #[schema(example = "2025-07-18T17:22:00+04:00")]
46     pub created_at: String,
47     /// The date and time when the blog post was last updated.
48     #[schema(example = "2025-07-18T17:22:00+04:00")]
49     pub updated_at: String,
50     /// Whether the blog post is publicly visible.
51     #[schema(example = true)]
52     pub public: bool,
53     /// Whether the blog post is published.
54     #[schema(example = true)]
55     pub published: bool,
56     /// The tags and labels associated with the blog post.
57     #[schema(example = "[ ]")]
58     pub tags: Vec<String>,
59     /// A list of comments associated with the blog post.
60     #[schema(example = "[ ]")]
61     pub comments: Vec<BlogComment>,
62     /// The amount of views the blog post has received.
63     #[schema(example = 0)]
64     pub views: usize,
65     /// A list of user IDs who have liked the blog post.
66     #[schema(example = "[ ]")]
67     pub likes: Vec<usize>,
68 }
```

```
18 #[derive(Default, Debug, Clone, Serialize, Deserialize, ToSchema)]
19 pub struct BlogComment {
20     /// A unique identifier for the comment.
21     #[schema(example = 1)]
22     pub id: usize,
23     /// The ID of the blog post the comment belongs to.
24     #[schema(example = 1)]
25     pub blog_post_id: usize,
26     /// The content of the comment.
27     #[schema(example = "This is a comment.")]
28     pub content: String,
29     /// The ID of the user who created the comment.
30     #[schema(example = 1)]
31     pub user_id: Option<usize>,
32     /// An optional field for an anonymous user.
33     #[schema(example = "")]
34     pub anonymous_user: Option<TicketAnonymousUser>,
35     /// The date and time when the comment was created.
36     #[schema(example = "2025-07-18T17:22:00+04:00")]
37     pub created_at: String,
38     /// The date and time when the comment was last updated.
39     #[schema(example = "2025-07-18T17:22:00+04:00")]
40     pub updated_at: String,
41     /// A list of user IDs who have liked the comment.
42     #[schema(example = "[ ]")]
43     pub likes: Vec<usize>,
44 }
45 }
```

Productization & Scale: Blog Engine

Edit Blog Post devcon-2025



The Developer's Conference, or [DevCon](https://conference.mscc.mu/) for short, is by far the largest public tech event in Mauritius. It is an event that happens once a year spanning 3 days. There are multiple simultaneous sessions happening at once where speakers from a vast range of backgrounds and often representing the companies at which they work, share knowledge on some of their favorite technologies. During those 3 days, you learn all about industry trends, best practices and innovations and you get the chance to interact with very skilled individuals for networking purposes or drop by one of the booths of some companies that are sponsoring the event in order to maybe find a new job. Another big perk of attending the [DevCon](https://conference.mscc.mu/), at least for me and for dozens of my peers, is to get goodies which can be in the form of stickers, USB hubs, snacks, etc. For me personally, let's say I was there for all of the reasons I mentioned. The venue for this year was at Voilà Hotel at the Bagatelle mall, but for the last few [DevCon](https://conference.mscc.mu/)'s, it was actually hosted at the Caudan Arts Centre.

Day 1 - Thursday

During all 3 days, the sessions started at 09 00 and finished at 16 00. I woke up really early (which is very unnatural for me) in order to escape some traffic jams and to be sure to find a parking spot. I was actually surprised to be the first attendee arriving, and funnily enough, it was the case for the next 2 days as well (spoiler alert). I grabbed my badge and proceeded to do some networking and engage in some chat with a couple of the organizers. I met with some friends and past colleagues too, where we shared some updates about where we're at on a personal and on a professional level, and how life's been treating us.

The venue was in no time at all crowded with attendees and I had to squeeze in to get anywhere. In some of the sessions, there were even not enough seats and many attendees had no choice but to stand at the back. This came as no surprise, however, given the sheer amount of attendees who register for the [DevCon](https://conference.mscc.mu/) (2000+ this year alone), and given that the venue was far smaller than that of the previous [DevCon](https://conference.mscc.mu/)'s. As with every year, based on what I noticed, there is a larger share of university students compared to working professionals attending the [DevCon](https://conference.mscc.mu/). I don't mind this but the uni students usually deplete the supplies of goodies available, and most of them don't even attend many sessions or stick around for more than half a day.

Keynote: Framing the problem

The organizers of [DevCon](https://conference.mscc.mu/) welcomed all attendees, told us what the next 3 days would be about and shared the code of conduct. The opening keynote was by a manager working at [SWAN](https://www.swanforlife.com/en/about-swan/corporate-governance/swan-general-ltd), one of the largest insurance providers in Mauritius. The topic was about 'framing the problem', where interesting tips, techniques and case studies were shared in order to comprehend that sometimes taking a step back, looking at a problem at a higher level, questioning just about everything and challenging the way a process works can lead to new innovations and produce a better solution than when not thinking outside the box.

The manager also talked a lot about AI, and how [SWAN](https://www.swanforlife.com/en/about-swan/corporate-governance/swan-general-ltd) was embracing AI in their processes to solve real problems in their company. This included building flows with AI tools, automating whole processes and taking away as much as the manual work involved by _real humans_ since the latter is more prone to errors and can't work 24/7. [SWAN](https://www.swanforlife.com/en/about-swan/corporate-governance/swan-general-ltd) expressed how there is very likely an AI tool already available for

Enter Details



Kishan Takoordyal (@edgeking810)

14

DevCon 2025

devcon-2025

My favorite annual tech conference in Mauritius

devcon,2025,ai,security,bazel,microservices,keycloak,passport,browser

Add a preview image (optional)

https://api.kinesis.world/public/IMG_5652_Z5lws4oN.jpg

PICK

Carousel images (optional) +

Visibility

Private ☐ Public ☒

Everyone will be able to view this Blog Post.

Published ☒

UPDATE BLOG POST

VIEW BLOG POST

VIEW METRICS

DELETE BLOG POST

RESTORE PREVIOUS VERSION



Activated: Enterprise

ctrl / ⌘ + k

Productization & Scale: Backup Mechanism

New backup, backup schedule and content history components were introduced along with tests, routes, utils, docs and web pages.

```
15 #[derive(Default, Debug, Clone, Serialize, Deserialize, ToSchema)]
16 pub struct BackupSchedule {
17     /// A unique identifier for the backup schedule.
18     #[schema(example = 1)]
19     pub id: usize,
20     /// The name of the backup schedule.
21     #[schema(example = "Daily Backup")]
22     pub name: String,
23     /// Whether the backup schedule is enabled.
24     #[schema(example = true)]
25     pub enabled: bool,
26     /// The minute of the hour when the backup should be created.
27     #[schema(example = "0")]
28     pub minute: String,
29     /// The hour of the day when the backup should be created.
30     #[schema(example = "2")]
31     pub hour: String,
32     /// The day of the month when the backup should be created.
33     #[schema(example = "1")]
34     pub day_of_month: String,
35     /// The month of the year when the backup should be created.
36     #[schema(example = "1")]
37     pub month: String,
38     /// The day of the week when the backup should be created.
39     #[schema(example = "1")]
40     pub day_of_week: String,
41     /// The amount of hours to keep the backup.
42     #[schema(example = 24)]
43     pub expiry_hours: usize,
44 }
```

```
56 #[derive(Debug, Clone, Serialize, Deserialize, ToSchema)]
57 pub struct ContentHistory {
58     /// A unique identifier for the content history.
59     #[schema(example = 1)]
60     pub id: usize,
61     /// The type of content that this history entry is associated with.
62     #[schema(example = "DATA")]
63     pub content_type: ContentHistoryType,
64     /// The id of the content that this history entry is associated with.
65     #[schema(example = 1)]
66     pub content_id: usize,
67     /// The actual content of the history entry.
68     #[schema(example = "This is an example content history entry.")]
69     pub content: String,
70     /// The date and time when the content history was created.
71     #[schema(example = "2025-08-15T21:44:00+04:00")]
72     pub created_at: String,
73 }
```

```
16 #[derive(Default, Debug, Clone, Serialize, Deserialize, ToSchema)]
17 pub struct Backup {
18     /// A unique identifier for the backup.
19     #[schema(example = 1)]
20     pub id: usize,
21     /// The name of the backup.
22     #[schema(example = "example-backup.tar.gz")]
23     pub name: String,
24     /// The description of the backup.
25     #[schema(example = "This is an example backup description.")]
26     pub description: String,
27     /// The date and time when the backup was created.
28     #[schema(example = "2025-08-06T11:54:00+04:00")]
29     pub created_at: String,
30     /// An optional date and time when the backup should be deleted.
31     #[schema(example = "2025-08-07T11:55:00+04:00")]
32     pub expiry: Option<String>,
33 }
```

```
18 #[derive(Debug, Clone, Serialize, Deserialize, ToSchema, PartialEq)]
19 pub enum ContentHistoryType {
20     DATA,
21     ROUTE,
22     BLOG_POST,
23 }
```

BACKUP-20260208_0000_1.TAR.GZ SUN FEB 08 2026

Description: Scheduled backup for Daily at Midnight (1)

Expiry: 2/10/2026, 4:00:00 AM



CREATE A NEW BACKUP SCHEDULE

Enter a name... (e.g Daily Backup)

CONFIGURE SCHEDULE

Enter minute... (e.g 30)

Enter hour... (e.g 12)

Enter day of month... (e.g 1)

Enter month... (e.g *)

Enter day of week... (e.g *)

Enter number of hours to keep backups... (e.g 24)

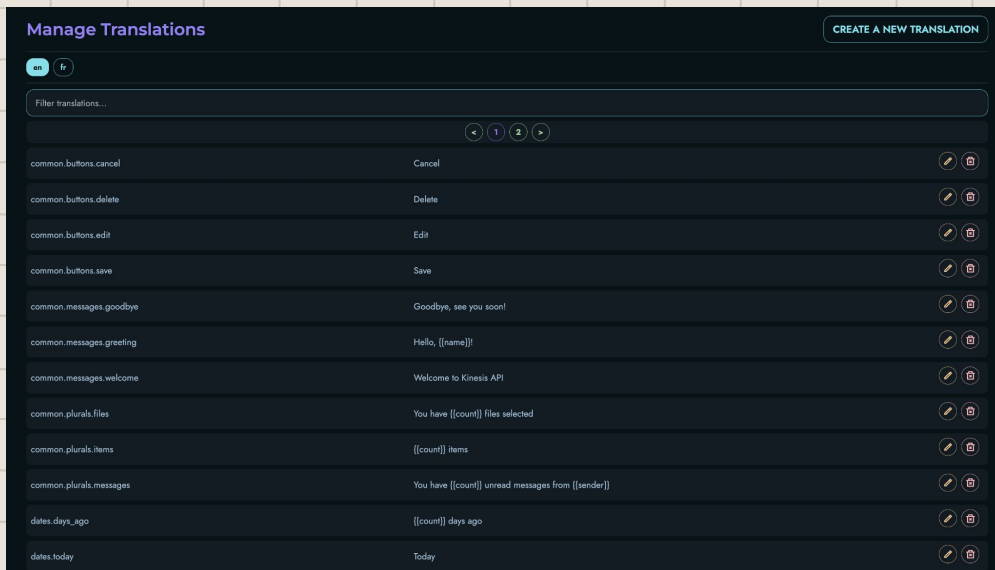
Is Enabled? ☒

CREATE

Productization & Scale: Localization Engine

New localization engine component for managing translations for different languages.

```
47  /// Localization engine that manages translations for multiple locales
48  #[derive(Debug)]
49  pub struct LocalizationEngine {
50      /// Translation data for all loaded locales
51      translations: RwLock<HashMap<Locale, HashMap<Key, TranslationValue>>>,
52      /// Metadata about loaded locale files for hot reloading
53      locale_metadata: RwLock<HashMap<Locale, LocaleMetadata>>,
54      /// Default locale to fall back to when translations are missing
55      fallback_locale: Locale,
56      /// Track missing translation keys for diagnostics
57      missing_keys: RwLock<HashSet<String>>,
58      /// Pluralization rules for different languages
59      plural_rules: HashMap<String, PluralRuleFn>,
60      /// Enable auto-reloading of translation files when they change
61      auto_reload: bool,
62      /// Cache for commonly used translations
63      translation_cache: RwLock<HashMap<(String, String, Option<u64>), String>>,
64  }
```



Productization & Scale: Search Engine

New search engine component for finding relevant data quicker.

```
12 pub fn fetch_records(engine: &DBEngine, table_name: &Str, encryption_key: &Str) -> Vec<Record> {
13     let mut all_records = Vec::<Record>::new();
14
15     let tables_with_encrypted_data: HashMap<&Str, Vec<&Str>> = HashMap::from([
16         ("config", vec!["value"]),
17         ("data_pair", vec!["value"]),
18         ("encryption_key", vec!["key"]),
19         ("snippet", vec!["content"]),
20     ]);
21
22     if let Some(table) = engine.get_table(table_name) {
23         for (_, record_lock) in &table.data {
24             if let Ok(record) = record_lock.read() {
25                 all_records.push(record.clone());
26             }
27         }
28     }
29
30     for record in all_records.iter_mut() {
31         for (k, v) in record.values.iter_mut() {
32             if let ValueType::Str(s) = v {
33                 let mut value = s.into_string();
34
35                 if value.starts_with("[BLOB REF:") && value.contains("size:") {
36                     value = engine.resolve_blob_reference(&value);
37                 }
38
39                 if let Some(encrypted_value) = tables_with_encrypted_data.get(table_name) {
40                     if encrypted_value.contains(&k.as_str()) {
41                         value = match EncryptionKey::decrypt(value, encryption_key) {
42                             Ok(v) => v,
43                             Err(_) => String::new(),
44                         };
45                     }
46                 }
47
48                 *v = ValueType::Str(StringValue::new(value));
49             }
50         }
51     }
52
53     all_records
54 }
```

```
9 fn fuzzy_match(target: &Str, query: &Str, max_dist: u8, substring: bool) -> (bool, usize) {
10     let mut best_score = 0;
11     let mut matched = false;
12
13     if substring {
14         let qlen = query.chars().count();
15         let tlen = target.chars().count();
16
17         if qlen > tlen {
18             let dist = levenshtein(&target, query);
19             if dist <= max_dist as usize {
20                 matched = true;
21                 best_score = 8 - dist; // higher score for closer match
22             }
23         } else {
24             for i in 0..=(tlen - qlen) {
25                 let window = target.chars().skip(i).take(qlen).collect::<String>();
26                 let dist = levenshtein(&window, query);
27                 if dist <= max_dist as usize {
28                     matched = true;
29                     let score = 10 - dist;
30                     best_score = best_score.max(score);
31                 }
32             }
33         }
34     } else {
35         let dist = levenshtein(&target, query);
36         if dist <= max_dist as usize {
37             matched = true;
38             best_score = 6 - dist; // lower weight for full-field fuzzy
39         }
40     }
41
42     (matched, best_score)
43 }
```

```
5 #[derive(Debug, Clone, Serialize, Deserialize, ToSchema)]
6 pub enum StringMatch {
7     Contains(String, bool), // (value, case_sensitive)
8     Equal(String, bool), // (value, case_sensitive)
9     StartsWith(String, bool), // (value, case_sensitive)
10    EndsWith(String, bool), // (value, case_sensitive)
11    Fuzzy(String, Option<u8>, bool), // (value, max edit distance, substring match)
12 }
```

```
6 #[derive(Debug, Clone, PartialEq)]
7 enum Token {
8     // Field identifiers
9     Identifier(String),
10     Dot,
11
12     // Operators
13     OpContains,
14     OpEquals,
15     OpStartsWith,
16     OpEndsWith,
17     OpFuzzy,
18     OpGreaterThan,
19     OpLessThan,
20     OpBetween,
21
22     // Logical operators
23     And,
24     Or,
25     Not,
26
27     // Values
28     StringLiteral(String),
29     NumberLiteral(f64),
30     BooleanLiteral(bool),
31
32     // Structure
33     LParen,
34     RParen,
35     Comma,
36 }
```

```
15 pub enum NumericMatch {
16     GreaterThan(f64),
17     LessThan(f64),
18     Equal(f64),
19     Between(f64, f64),
20 }
```

```
23 pub enum BooleanMatch {
24     Equal(bool),
25     NotEqual(bool),
26 }
```

Producttization & Scale : Licensing

Manage Plans

[CREATE A NEW PLAN](#)

1 | community | Active

Community

Perfect for learning and small projects

0 USD FLAT / month

0 USD FLAT / year

Created on: 10/27/2025

Last updated on: 10/27/2025

EDITDELETE

Features: +

- API Creation: Basic API routes (limit 10)
- Database Storage: 100 MB
- Media Storage: 2 GB
- Monthly Requests: 100K
- X Engine: Basic blocks only
- Analytics: Basic usage stats
- Backups: Manual only
- Authentication: Basic authentication
- Support: Community forum
- Additional Features: Self-hosting, OpenAPI documentation

2 | essential | Active

Essential

For growing applications

99 USD FLAT / month

1188 USD FLAT / year

Created on: 10/27/2025

Last updated on: 10/27/2025

EDITDELETE

Features: +

- API Creation: Unlimited API routes
- Database Storage: 5 GB
- Media Storage: 50 GB
- Monthly Requests: 5M
- X Engine: Standard blocks
- Analytics: Standard analytics
- Backups: Daily backups (7-day retention)
- Authentication: Multiple authentication options
- Support: Email support (48-hour response)
- Additional Features: Self-hosting, OpenAPI documentation, Webhooks, Basic localization, Email templates

3 | professional | Active

Professional

For professional teams

499 USD FLAT / month

5988 USD FLAT / year

Created on: 10/27/2025

Last updated on: 10/27/2025

EDITDELETE

Features: +

- API Creation: Unlimited API routes
- Database Storage: 20 GB
- Media Storage: 250 GB
- Monthly Requests: 25M
- X Engine: All advanced blocks
- Analytics: Advanced analytics + reporting
- Backups: Hourly backups (30-day retention)
- Authentication: Advanced authentication + SSO
- Support: Priority support (12-hour response)
- Additional Features: Self-hosting, OpenAPI documentation, Advanced Webhooks, Full localization, Email templates including customization, Content versioning

4 | enterprise | Active

Enterprise

For large organizations

50 USD PER_USER / month

600 USD PER_USER / year

Created on: 10/27/2025

Last updated on: 10/27/2025

EDITDELETE

Features: +

- API Creation: Unlimited API routes
- Database Storage: Unlimited
- Media Storage: Unlimited
- Monthly Requests: Unlimited
- X Engine: Custom block development
- Analytics: Custom analytics & BI integration
- Backups: Custom retention policies
- Authentication: Custom authentication integration
- Support: Dedicated support representative + SLA
- Additional Features: Self-hosting, OpenAPI documentation, Advanced Webhooks, Full localization, Email templates including customization, Content versioning, Custom integrations, On-premise deployment, Training & compliance, Compliance documentation

```
44 pub struct LSPlan {
45     pub id: i32,
46     pub name: String,
47     pub display_name: String,
48     pub description: Option<String>,
49     pub created_at: String,
50     pub updated_at: String,
51     pub price_monthly: f64,
52     pub price_yearly: Option<f64>,
53     pub billing_type: String,
54     pub currency: String,
55     pub active: bool,
56 }
```

```
27 pub struct LSLicense {
28     pub id: i32,
29     pub plan_id: i32,
30     pub owner_id: i32,
31     pub coupon_id: Option<i32>,
32     pub name: String,
33     pub license_key: String,
34     pub max_instances: Option<i32>,
35     pub max_users: Option<i32>,
36     pub issued_at: String,
37     pub expires_at: String,
38     pub status: String,
39     pub billing_period: String,
40 }
```

```
69 pub struct LSPlanFeature {
70     pub id: Option<i32>,
71     pub plan_id: i32,
72     pub feature_id: String,
73     pub feature_name: String,
74     pub feature_details: String,
75 }
```

```
60 pub struct LSPriceInfo {
61     pub original_amount: f64,
62     pub discount_amount: f64,
63     pub final_amount: f64,
64     pub currency: String,
65 }
```

```
16 pub struct LSActivation {
17     pub id: i32,
18     pub license_id: i32,
19     pub instance_id: String,
20     pub started_at: String,
21     pub last_check_in: String,
22     pub status: String,
23 }
```

Productization & Scale : Licensing

Manage License

.....

VALIDATE LICENSE

DEACTIVATE LICENSE

Plan Details	Plan Features	License Details	Activation Status
1 community Active Community Perfect for learning and small projects 0 USD FLAT / month 0 USD FLAT / year Created on: 10/27/2025 Last updated on: 10/27/2025	• API Creation: Basic API routes (limit 10) • Database Storage: 100 MB • Media Storage: 2 GB • Monthly Requests: 100K • X Engine: Basic blocks only • Analytics: Basic usage stats • Backups: Manual only • Authentication: Basic authentication • Support: Community forum • Additional Features: Self-hosting, OpenAPI documentation	6 COMMUNITY Coupon: N/A COMMUNITY Status: ACTIVE Max Instances: 1 Max Users: 1 Billing Period: YEARLY Original Amount: 0 USD Discounted Amount: 0 USD Final Amount: 0 USD Issued At: 12/7/2025 Expires At: 12/12/2027	108 1lj9-ph8dG-Hd281-1CS08-XtYih COMMUNITY Status: ACTIVE Started At: 1/10/2026, 4:18:15 PM Last check-in: 2/8/2026, 9:50:00 PM

```
268 #[derive(Default, Clone, Debug, ToSchema)]
269 pub struct License {
270     pub license_key: String,
271     pub activation_id: Option<i32>,
272     pub instance_id: String,
273     pub instance_endpoint: String,
274     pub license_token: Option<String>,
275 }
```

```
253 #[derive(Clone, Debug, Serialize, Deserialize, ToSchema)]
254 pub struct LicenseBackupPolicy {
255     pub backup_enabled: bool,
256     pub backup_schedule_enabled: bool,
257     pub backup_schedule_limit_daily: bool,
258     pub backup_schedule_limit_hourly: bool,
259     pub backup_schedule_max_retention: Option<usize>,
260 }
```

Productization & Scale : Licensing

- Unified License Server (license.kinesis.world) handles all licensing operations across instances.
- License Server Components: User, Config, Plan, PlanFeature, Coupon, License, Activation.
- Single license key can activate multiple Kinesis API deployments with unique instance tracking.
- Plans determine available X Engine blocks, storage limits, backup policies, and custom features.
- License key validation, JWT token issuance, and encrypted storage of credentials.
- Background check-ins every 5 minutes ensure license remains active and compliant.
- License Server tracks active users per instance and enforces plan-based user limits.
- Automatic JWT refresh when tokens expire to maintain continuous validation.
- Failed validation reverts instance to Community tier features without data loss.
- License registration, validation, and deactivation restricted to ROOT users via /web/license.
- Feature gating applied at route execution level through X Engine block availability checks.
- License removal notifies License Server and safely reverts to base feature set.
- License validation happens automatically; users experience uninterrupted feature access when compliant.
- License Server URL is obfuscated using ChaCha20-Poly1305 encryption with scrambled key parts embedded in the binary to prevent endpoint discovery and ensure secure communication.

Productization & Scale: Other changes

- New workflow for building multi-arch & multi-registry docker images through docker buildx.
- API routes for warming up Kinesis DB and compacting records.
- New SimpleUpdateBlock for the X Engine.
- Video tutorials on YouTube and Facebook/Instagram socials.
- Removal of the publish functionality for data objects.
- Making min/max fields optional for structures.
- Addition of the migration commands section in changelog entries.
- Allow custom locators for redirects.
- Asynchronously send emails in another thread.
- CI/CD pipeline improvements.
- Scalar API documentation improvements.
- Improvements to pagination logic and to SEO for blog posts.

```
10 #[derive(Default, Debug, Clone, Serialize, Deserialize, PartialEq, ToSchema)]
11 pub struct SimpleUpdateBlock {
12     /// The global index of the block.
13     #[schema(example = 0)]
14     pub global_index: u32,
15     /// The block or local index of the block.
16     #[schema(example = 0)]
17     pub block_index: u32,
18     /// The x position of the block when rendering it on the Web UI.
19     #[schema(example = 0)]
20     pub x: f64,
21     /// The y position of the block when rendering it on the Web UI.
22     #[schema(example = 0)]
23     pub y: f64,
24     /// The collection being referenced.
25     #[schema(example = "users")]
26     pub ref_col: String,
27     /// The property to update for all data records that match the 'targets' field of this block. If left empty, the whole data record will be replaced by a new one.
28     #[schema(example = "name")]
29     pub ref_property: String,
30     /// Whether to save the results of the update.
31     #[schema(example = true)]
32     pub save: bool,
33     /// A list of targets in order to decide which data records should be updated.
34     pub targets: Vec<UpdateTarget>,
35     /// The data to set for the property of data records not having been filtered out or to replace the data record by.
36     pub set: RefData,
37 }
```

```
1  #!/bin/bash
2
3  set -e # Exit immediately if a command exits with a non-zero status
4
5  # Extract the version from Cargo.toml
6  VERSION=$(grep "^version" ./Cargo.toml | head -1 | cut -d '"' -f2)
7
8  # Ensure the buildx builder exists and supports multi-arch builds
9  if ! docker buildx inspect multiarch &>/dev/null; then
10     echo "Creating a multi-architecture builder..."
11     docker buildx create --name multiarch --driver docker-container --use \
12         --buildkitd-flags '--debug' \
13         --driver-opt image=moby/buildkit:latest,network=host \
14         --driver-opt env.BUILDKIT_MEM_SWAPPINESS=0 \
15         --driver-opt env.BUILDKIT_MEMORY=8192m
16 fi
17
18 echo "Building and pushing version '${VERSION}'"
19 TAG=${VERSION} docker buildx bake --builder multiarch all
20
21 # Retag the image and push
22 echo "Building and pushing 'latest' tag"
23 docker buildx imagetools create \
24     gitea.konnect.dev/rust/kinesis-api:${VERSION} \
25     --tag gitea.konnect.dev/rust/kinesis-api:latest
26
27 docker buildx imagetools create \
28     docker.io/edgeking8100/kinesis-api:${VERSION} \
29     --tag docker.io/edgeking8100/kinesis-api:latest
30
31 echo "Building completed successfully!"
```



MULTI-DB & ANALYTICS

2026

Multi-DB & Analytics : External DB Support

- Multiple Database Backends: Support for Kinesis DB (default), SQLite, MySQL, and PostgreSQL with seamless switching.
- Diesel ORM Integration: Full Diesel support for all external databases with automatic migration management on startup.
- Bidirectional Data Migration: Migrate data between Kinesis DB and external databases in both directions.
- Component Conversion Layer: Universal converters for all components that handle encryption, blob references, and transactional consistency.
- Safe Migration Process: Lock files prevent concurrent migrations; checksums verify data integrity; detailed logging tracks progress.
- Batch Processing: Records processed in chunks to minimize memory overhead; streaming inserts for large datasets..
- Transaction Atomicity: Each table migration wrapped in transaction; rollback on any error prevents partial data transfers.
- Error Recovery: Graceful degradation with connection retry logic, partial data cleanup, and diagnostic logging for troubleshooting.
- Environment Variables: Configure via DB_BACKEND, DATABASE_URL, DB_NAME, and connection pool settings.
- Schema Compatibility: Automatic backend detection with SQL migrations supporting all three external database syntaxes.
- Migration Tests: Comprehensive unit tests, round-trip testing, and crash recovery validation across all backends.
- Future Enhancements: Scheduled migrations, incremental sync, multi-destination export, and query optimization per backend.

Multi-DB & Analytics: Email Templates

A new component was introduced along with tests, routes, utils, docs and web pages for allowing end-users to create reusable email templates across various operations.

VIEW EMAIL TEMPLATE

RESTORE PREVIOUS VERSION

Email Body (HTML)

Rendered Email Body (Preview) 🌙

```
<meta name="color-scheme" content="dark">
<meta name="supported-color-schemes" content="dark">
<div style="background-color: oklch(18% 0.019 237.69); padding: 3em 2em 3em 2em; border-radius: 16px;">
  <h1 style="color: #9582F2; margin-bottom: 8px;">Hey there, {name} 👋!</h1></div>

  <p style="color: #06D5B3; margin-top: -4px;">A password reset has been requested for your account on
    <a href="{site_url}" style="color: #89e0eb;">{site_name}</a> 🌟.</p>

  <p style="color: #06D5B3; margin-top: -8px;">Please use the above link in order to change your password or alternatively, copy and paste the below url in your browser 📄.</p>
  <p style="color: #06D5B3; margin-top: -8px;">Kindly disregard if you did not request a password change.</p>

  <p style="color: #89e0eb; margin-top: -4px;">{site_url}</p>

  <p style="color: #ffbbbd; margin-top: 10px;">This is an automatic response generated when a password change is requested. Please do not reply to this email.</p>
</div>
```

Hey there, {name} 👋!

A password reset has been requested for your account on {site_name} 🌟.

Please use the above link in order to change your password or alternatively, copy and paste the below url in your browser 📄.

Kindly disregard if you did not request a password change.

{site_url}

This is an automatic response generated when a password change is requested. Please do not reply to this email.

Properties

Default Variables +

Name

forget_password

Purpose

Password reset email template

Subject

Reset your password for {site_name}

Deletable ?

☐

name

John Doe

×

site_url

https://api.kinesis.world/web/forget?user_id={user_id}&reset_token={reset_token}

×

site_name

Kinesis API

×

Multi-DB & Analytics: Vault

A new component was introduced along with tests, routes, utils, docs and web pages for allowing end-users to create and use secrets in their routes.

Manage Vault in Konnect - Social Media

ID: 1 | konnect

A next-gen social media.

/api/v2/konnect

1 MEMBER

VIEW MEMBERS

VAULTS (3)

CREATE NEW

Filter vault entries...

< 1 >

5

SECRET_KEY_KDB

.....



4

SECRET_KEY_SQLITE

.....



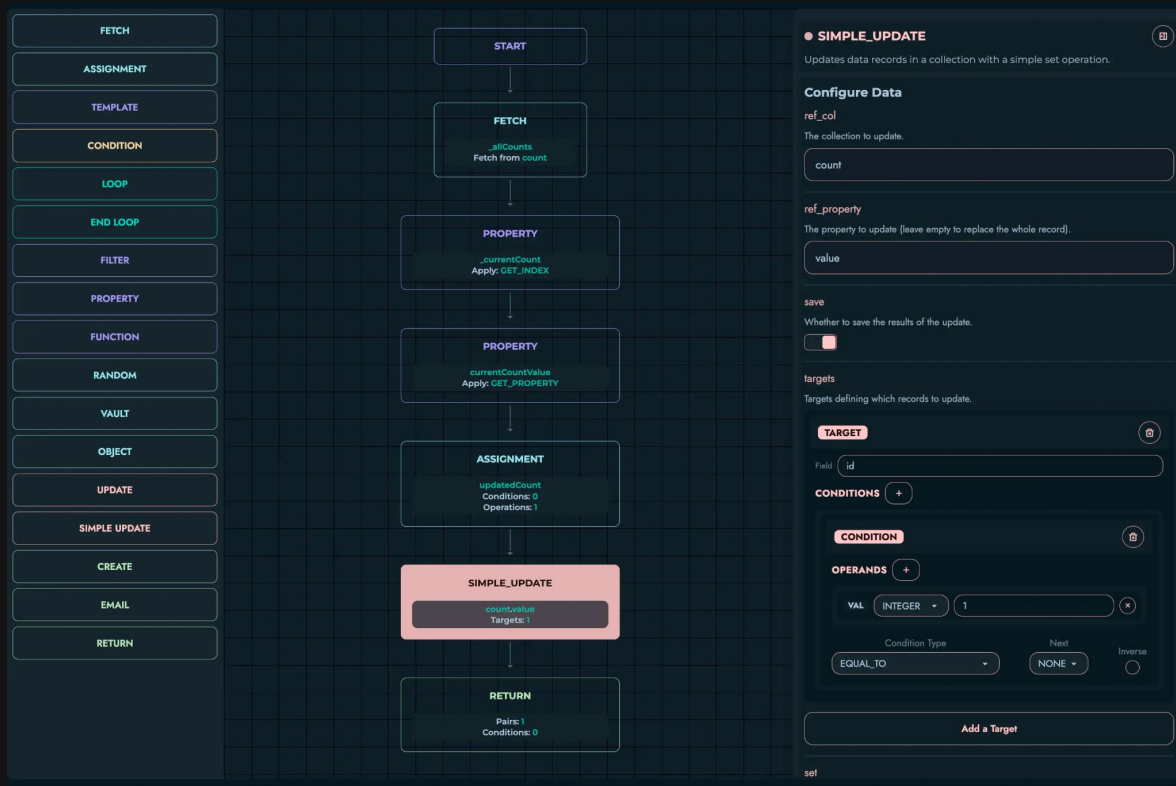
1

SECRET_KEY

.....



Multi-DB & Analytics : X Engine UI Refactoring



Multi-DB & Analytics : New X Engine Blocks

● RANDOM

Generates a random value within a specified range.

Configure Data

local_name

The variable name to store the random value.

Enter local_name...

min

The lower bound of the random value.

VAL STRING Value...

max

The upper bound of the random value.

VAL STRING Value...

data_type

The type of random value to generate.

INTEGER

● VAULT

Retrieves a secret value from the vault by key.

Configure Data

local_name

The variable name to store the retrieved vault value.

Enter local_name...

key

The key of the vault entry to retrieve.

Enter key...

● EMAIL

Sends an email using a named template.

Configure Data

email_template_name

The name of the email template to use.

Enter email_template_name...

subject

The subject/title of the email (optional, leave empty to use the template default).

Not set -- defaults to 1 Set

variables

Variables to substitute in the email template.

Add a Pair

recipients

The email addresses of the recipients.

Add an Item

send_as

The sender email address (optional).

Not set -- defaults to 1 Set

send_individually

Whether to send individually to each recipient rather than as one email.

☐

send_async

Whether to send the email asynchronously.

☐

conditions

Conditions to check before sending the email.

Add a new Condition

Multi-DB & Analytics : Route Templates

Manage Route Templates **set globally**

ROUTE TEMPLATES (3)

Filter route templates...

< 1 >

GENERATE NUMBERS



ID: 9 **GET**

A template for generating some random numbers.

COUNT ITEMS



ID: 8 **GET**

A template for counting and returning the amount of items in a collection.

HELLO WORLD



ID: 7 **GET**

Just a simple personalized hello world message.

Variables

Field name	Description / Purpose	Default Value (optional)	
amount	Amount of IDs to generate.	10	×
min	The minimum size for an ID.	1	×
max	The maximum size for an ID.	100	×
+ Add Variable			

- Introduce the Route Template component for defining reusable, parameterized X Engine route blueprints.
- Add Route Template Variable support for declaring named placeholders with descriptions and optional default values.
- Implement template application workflow in the route creation page: browse templates, fill in variable values, and populate the form.
- Implement full variable substitution across all X Engine block types when applying a route template.

Multi-DB & Analytics : Instance Manager

- Single Interface for managing instances and licenses.
- Ability to spin and deploy Kinesis API instances within seconds, including DNS configuration.
- Real-time metrics on all instances.



IM

Dashboard

Instances

Licenses

Managed

Users

Configs

Events



WELCOME BACK

Hi, Kishan 🙌

Here's everything you need to manage your instances.

Settings

Profile

YOUR TOOLS



Settings

Edit user information



Instances

View all API instances



Licenses

View license servers



Managed

Managed instances

PLATFORM ADMINISTRATION

Admin



Users

Manage all users



Configs

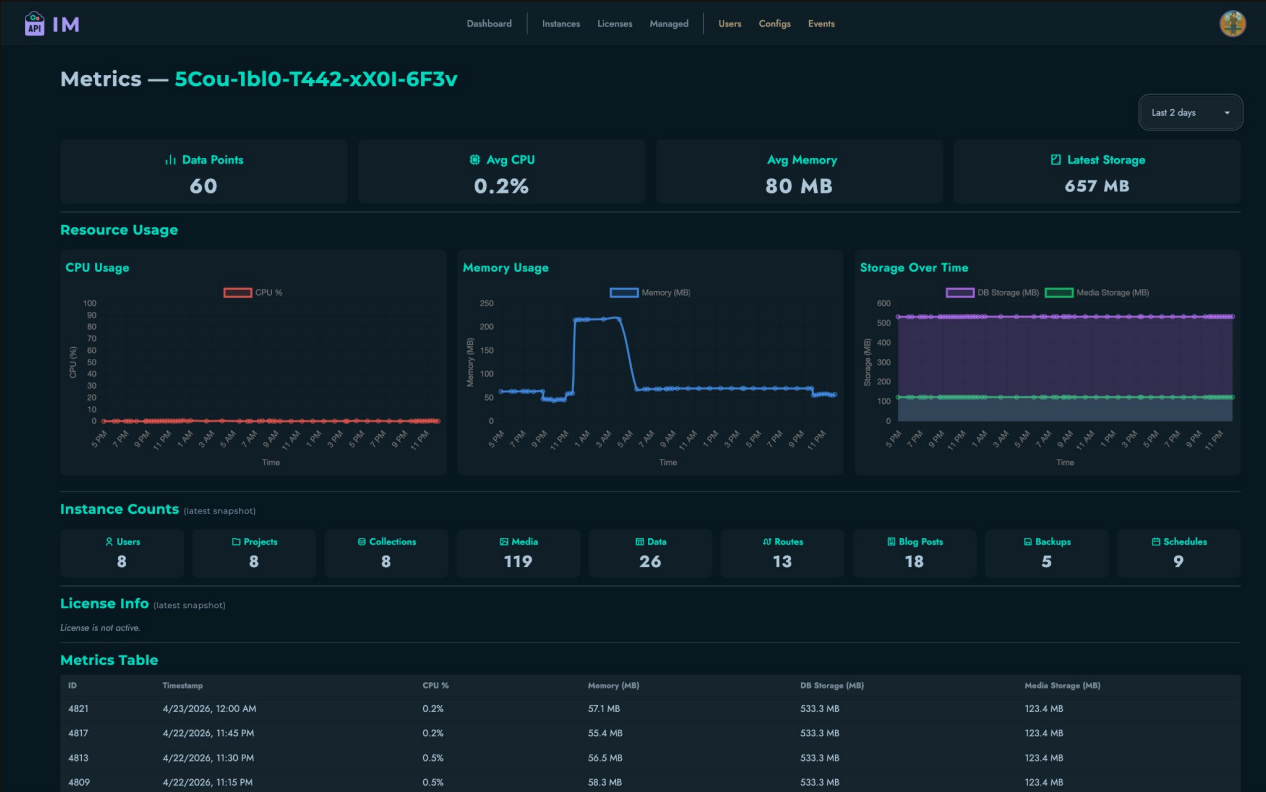
Platform configuration



Events

View all events

Multi-DB & Analytics : Instance Manager



CREATE MANAGED INSTANCE

Name

Description

Domain (e.g. myinstance.example.com)

Version (e.g. latest)

CREATE

Kinesis API Beta

Managed

beta.kinesis.world

Beta Instance for well... beta releases

<> latest 4/12/2026, 10:52:50 PM

T Name

Desc

<> Version

Domain

Delete

API

🏠

📁

📅

📊

📋

📌

📝

🔍

🔗

🔧

🔖

🔔

Planned Tasks - 1.0 Release

Tracking the remaining tasks for 2026 on Kinesis API.

📅 Created: Jun 10, 2026, 3:31 PM ⚡ Updated: Jun 10, 2026, 3:59 PM

PRIVATE

👁 Watching

👤 Members (4)

🕒 Activities

🔔 Notifications 5

Settings ⌵

To Do

🖼️

Feature Important

Integrated AI Assistant for Routes

📅 Jul 15

+ Add Card

📅 Jun 10, 2026, 3:36 PM

📅 Jun 10, 2026, 3:43 PM

In Progress

🖼️

Feature Improvement

Workflow Boards

📅 Jun 8

+ Add Card

📅 Jun 10, 2026, 3:36 PM

📅 Jun 10, 2026, 3:37 PM

Resolved

🖼️

Feature Improvement

Multi-Database Support

📅 Jun 10

+ Add Card

📅 Jun 10, 2026, 3:36 PM

📅 Jun 10, 2026, 3:38 PM

Closed / Discarded

🖼️

X Engine

OAuth2 Authentication Block

📅 Jun 10

+ Add Card

📅 Jun 10, 2026, 3:36 PM

📅 Jun 10, 2026, 3:39 PM

+ New List

👤

Activated: Enterprise

ctrl / ⌘ + k

Planned Tasks - 10 Release

Tracking Info

Created: Jun 10, 2026, 3:42 PM

PRIVATE

To Do

Feature

Important

DESCRIPTION

An AI assistant feature needs to be implemented in Routes to help with developing routes on the X Engine.

START DATE

15 / 6 / 2026

DUE DATE

15 / 7 / 2026

RECURRING

Weekly

CUSTOM FIELDS

Priority

P2

Tentative due date

31 / 7 / 2026

Story Points

21

CHECKLISTS

+ Add

Actions

1/3

☒ Research

D

M

Y

☐ Development

D

M

Y

☐ Integration with existing providers

D

M

Y

Add item...

ATTACHMENTS

+ Upload

@ URL

plan.webp

16.1 KB · 1 minute ago

MEMBERS

Kishan Test

Kishan Games

Kishan iCloud

Kishan Takoordyal

LABELS

Feature

Improvement

Bug

Important

COVER IMAGE

/public/pngtree-futuristic-ai-technology-source-and-related-cont

Set

COVER COLOR

Clear

Watching

Archive

Delete Card

Created: Jun 10, 2026, 3:42 PM

Updated: Jun 10, 2026, 3:50 PM

Multi-DB & Analytics : AI Assistant

● AI Assistant

Use the AI Assistant to help generate, analyze, explain, and recommend improvements for your API flows.

Generate

Analyze

Explain

Recommend

Generate a flow for fetching and returning all records from the "creatures" collection where their "is_pet" property is equal to a parameter passed by the end user.

🔧 Generate

Route Suggestion:

ID: fetch_creatures_by_pet_status
Path: /creatures/filter
Method: GET

```
{
  "assignment_blocks": [],
  "condition_blocks": [],
  "create_blocks": [],
  "email_blocks": [],
  "end_loop_blocks": [],
  "fetch_blocks": [
    {
      "id": "fetch_creatures_by_pet_status",
      "method": "GET",
      "path": "/creatures/filter",
      "status": 200
    }
  ]
}
```

📄 Import to Canvas

● AI Assistant

Use the AI Assistant to help analyze and explain your API flows.

Analyze

Explain

Get a plain-English summary of what this flow does.

📄 Explain Current Flow

This API retrieves a list of species from the "creatures" collection, filtered by their pet status. Here's how it works:

- Fetch Data**: The system first grabs every record from the "creatures" table and stores them in a temporary list called "_allCreatures".
- Initialize List**: It prepares an empty list called "_foundCreatures" to hold the results that match your search.
- Loop Through Creatures**: The system looks at every individual creature in the list one by one.
- Extract Information**: For each creature, it identifies its species and checks whether it is a pet (the "is_pet" status).
- Compare and Filter**: It compares the creature's "is_pet" status against the value you provided in your request ("pet").
- Collect Matches**: If the creature matches your criteria, the system takes that creature's species name and adds it to our results list.
- Return Results**: Finally, the API sends back a response containing only the names of the species that matched your filter.

Important Notes:

- This is a "GET" request located at the "/fetch" path.
- The "pet" parameter in your request determines whether you are looking for pets (true) or non-pets (false).

● AI Assistant

Use the AI Assistant to help generate, analyze, explain, and recommend improvements for your API flows.

Generate

Analyze

Explain

Recommend

Make this flow more efficient.

💡 Get Recommendations

FILTER_BLOCKS

+ Add

Currently, the flow fetches all creatures and iterates through every single one in the database. Adding a filter block immediately after fetching will allow you to reduce the loop's iteration count significantly by only processing records where 'is_pet' is true.

CONDITION_BLOCKS

+ Add

To improve the user experience, a condition block should be added before the return to check if any pets were found. This allows for a custom message or 'No results' status when the result list is empty.

NEXT STEPS

Productivity

Integrated task planning and organization system with project tracking, deadline management, and team collaboration tools. Enhanced analytics, performance metrics, and reporting dashboards to monitor productivity and identify bottlenecks across workflows.

Extensibility

New X Engine blocks (Authentication) for advanced workflow automation. Webhook support and dynamic WebSocket connections for real-time event handling and bidirectional communication between systems.

Experience

Associated SDK or module for JavaScript frameworks (React, Vue, Angular) enabling seamless integration and component-based development.

THANK YOU!

"Every complex system starts with a single decision and the willingness to learn from mistakes along the way."

