

# Jet AI Project Documentation

## Getting Started

1. Clone the repo: <https://github.com/ambroquach29/jetai-project>

## Setting up Backend

2. Navigate to the **/backend** directory
3. Run **npm install** to install server dependencies
4. Create a **.env** file with these values (also refer to **.env.example**)

```
5. PORT=5000
6. ACCESS_TOKEN=apollo-starter-kit
7. HASURA_URL="http://localhost:8080/v1/graphql"
8. HASURA_ADMIN_SECRET=
9. POSTGRES_URL="postgresql://postgres:postgrespassword@localhost:5432/postgres?schema=public"
```

10. Set up Postgres database and Hasura GraphQL Engine running as Docker containers using Docker Compose. The **docker-compose.yml** has the all the configurations needed.
11. Prereq: We need [Docker Desktop](#) working on your machine
12. Run **docker compose up -d** to start the containers

```
> docker compose up -d
[+] Running 3/3
  ✓ Container backend-postgres-1          Running
  ✓ Container backend-data-connector-agent-1 Healthy
  ✓ Container backend-graphql-engine-1     Running
> docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
787c0ca764a9   postgres:15         "docker-entrypoint.s..."   37 hours ago       Up 37 hours        0.0.0.0:5432
32--5432/tcp    backend-postgres-1
27b5ac14e091   hasura/graphql-engine:v2.37.0   "/bin/sh -c '$(HGE_...'"  38 hours ago      Up 37 hours (healthy)  0.0.0.0:80
80--8080/tcp    backend-graphql-engine-1
e60609f297c6   hasura/graphql-data-connector:v2.37.0  "/app/run-java.sh"   38 hours ago      Up 37 hours (healthy)  5005/tcp,
0.0.0.0:8081->8081/tcp  backend-data-connector-agent-1

```

13. Run **docker ps** to check its status
14. Now we've done setting up a local Postgres database connected Hasura GraphQL Engine - allow us to communicate with DB via GraphQL queries or REST
15. Initialize Postgres db using Prisma ORM
16. Navigate to **/backend/src/prisma**

17. Run **npx prisma migrate dev --name init\_jet\_entity** - This command creates and applies timestamped migration script(s) to our SQL db based on the model(s) defined inside schema.prisma

```
> npx prisma migrate dev --name init_jet_entity
Environment variables loaded from ../../.env
Prisma schema loaded from prisma/schema.prisma
Datasource "db": PostgreSQL database "postgres", schema "public" at "localhost:5432"

Applying migration `20240313183221_init_jet_entity`

The following migration(s) have been created and applied from new schema changes:

migrations/
└─ 20240313183221_init_jet_entity/
    └─ migration.sql

Your database is now in sync with your schema.

✓ Generated Prisma Client (v5.11.0) to ../../node_modules/@prisma/client in 104ms

  └── ~ / De/jetai-project/backend/s/schema └── master !3 ?2 ━ 4s ✘
```

18. We can use PgAdmin4 to verify the state of our DB, Prisma also provides a *migrations* table that keeps record of our migration history

The screenshot shows the PgAdmin4 interface with the following details:

- Left Panel (Browser):** Shows the Prisma schema structure under the "Schemas" section. It includes:
  - hdb\_catalog**
  - public** (selected):
    - Aggregates
    - Collations
    - Domains
    - FTS Configurations
    - FTS Dictionaries
    - FTS Parsers
    - FTS Templates
    - Foreign Tables
    - Functions
    - Materialized Views
    - Operators
    - Procedures
    - Sequences
  - Tables (2)** (selected):
    - Jet** (selected):
      - Columns
      - Constraints
      - Indexes
      - RLS Policies
      - Rules
      - Triggers
      - \_prisma\_migrations
      - Trigger Functions
      - Types
      - Views
  - Subscriptions**
  - Login/Group Roles**
  - Tablespaces**
- Right Panel (SQL Tab):** Displays the generated SQL code for the "Jet" table migration. The code is as follows:

```

-- Table: public.Jet
-- DROP TABLE IF EXISTS public."Jet";
CREATE TABLE IF NOT EXISTS public."Jet"
(
    id integer NOT NULL DEFAULT nextval('"Jet_id_seq"::regclass),
    name text COLLATE pg_catalog."default" NOT NULL,
    wingspan double precision NOT NULL,
    "numberOfEngines" integer NOT NULL,
    "manufacturingYear" integer NOT NULL,
    "createdAt" timestamp(3) without time zone NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "updatedAt" timestamp(3) without time zone NOT NULL DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT "Jet_pkey" PRIMARY KEY (id)
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Jet"
    OWNER to postgres;

```

```

1 SELECT * FROM public._prisma_migrations
2 ORDER BY id ASC

```

	finished_at	migration_name	logs	rolled_back_at	started_at	applied_steps_count
1	2024-03-13 18:32:21.746078+00	20240313183221_init_jet_entity	[null]	[null]	2024-03-13 18:32:21.723019+00	1

19. Navigate back to **/backend** directory and run **npm run dev** to start the Apollo server

```

> npm run dev
> jetai-backend@1.0.0 dev
> ts-node-dev src/server.ts

[INFO] 01:35:22 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.2, typescript ver. 5.4.2)
(node:67373) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)

⚡ Server is running!
⚡ Listening on port 5000
⚡ Query at http://localhost:5000/graphql

```

20. Go to <http://localhost:8080> on your browser, under the **Data** tab connect to Database via URL `postgres://postgres:postgrespassword@postgres:5432/postgres`

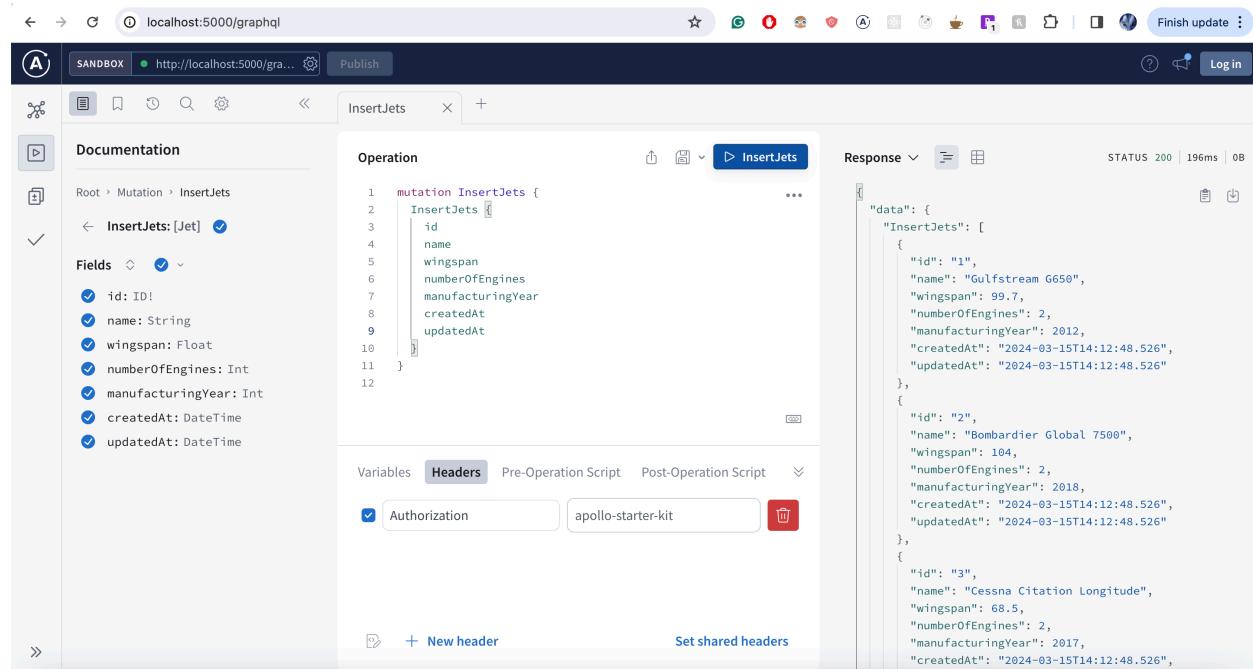
The screenshot shows the Hasura Data Manager interface at <http://localhost:8080/console/data/v2/manage/database/add?driver=postgres>. The 'DATA' tab is selected. A modal window titled 'Connect Postgres Database' is open, showing the 'Connection Details' section. The 'Database name' field contains 'jetdb'. Under 'Connect Database via', the 'Database URL' option is selected, and the URL 'postgres://postgres:postgrespassword@postgres:5432/postgres' is entered. A warning message states: 'This option exposes sensitive information such as password and hostname in your metadata as plaintext. The recommended way of adding connections is using an Environment variable.' Below the URL input, there are 'Advanced Settings' and 'GraphQL Customization' sections. At the bottom right of the modal is a 'Connect Database' button.

The screenshot shows the Hasura Data Manager interface at <http://localhost:8080/console/data/jetdb/schema/public>. The 'DATA' tab is selected. A modal window titled 'localhost:8080 says' asks 'Are you sure?' and states 'This will expose all the listed tables/views over the GraphQL API'. It has 'Cancel' and 'OK' buttons. The main interface shows the 'Data Manager' sidebar with 'Databases (1)' containing 'jetdb' and 'public'. The 'public' schema is selected, showing 'No tables or views in this schema'. The 'SQL', 'Native Queries', and 'Model Summary' sections are also visible.

21. After you connected to Postgres on Hasura, you will see something like the screenshot above. Click on **Track All**.

22. Go to <http://localhost:5000/graphql> and run the InsertJets query as shown below to populate our DB



```
1 mutation InsertJets {
2   InsertJets [
3     {
4       id: "1",
5       name: "Gulfstream G650",
6       wingspan: 99.7,
7       numberofEngines: 2,
8       manufacturingYear: 2012,
9       createdAt: "2024-03-15T14:12:48.526",
10      updatedAt: "2024-03-15T14:12:48.526"
11    },
12    {
13      id: "2",
14      name: "Bombardier Global 7500",
15      wingspan: 104,
16      numberofEngines: 2,
17      manufacturingYear: 2018,
18      createdAt: "2024-03-15T14:12:48.526",
19      updatedAt: "2024-03-15T14:12:48.526"
20    },
21    {
22      id: "3",
23      name: "Cessna Citation Longitude",
24      wingspan: 68.5,
25      numberofEngines: 2,
26      manufacturingYear: 2017,
27      createdAt: "2024-03-15T14:12:48.526",
28      updatedAt: "2024-03-15T14:12:48.526"
29    }
30  ]
31}
```

The screenshot shows the Apollo GraphQL playground interface. On the left, there's a sidebar with 'Documentation' and a tree view of the schema. The main area has tabs for 'Operation', 'Variables', 'Headers', 'Pre-Operation Script', and 'Post-Operation Script'. Under 'Headers', there's a checked checkbox for 'Authorization' and a dropdown set to 'apollo-starter-kit'. The 'Operation' tab contains the GraphQL mutation code. To the right, the 'Response' tab shows the JSON data returned by the mutation, which creates three jets with the specified details.

23. Here are the links to other GraphQL Queries and Mutations that you can run on the GraphQL API (Headers and Variables included): [QueryJets](#), [InsertJet](#), [UpdateJet](#), [DeleteJet](#)

## Setting up LLM

24. Navigate to the **/llm** directory

25. Run **source llm-env/bin/activate** to activate the virtual environment

26. Replace **openai\_api\_key** in **main.py** with my OpenAI key below

```
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
```

27. Run **uvicorn main:app --reload** to start the running our LLM service using Fast API -

Check <http://localhost:8000/docs> for Swagger endpoint specifications

```

> uvicorn main:app --reload
INFO:     Will watch for changes in these directories: ['/Users/ambroquach/Desktop/jetai-project/llm']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [85898] using StatReload
/Users/ambroquach/Desktop/jetai-project/llm/llm-env/lib/python3.12/site-packages/langchain/llms/_init__.py:548: LangChainDeprecationWarning: Importing LLMs from langchain is deprecated. Importing from langchain will no longer be supported as of langchain==0.2.0. Please import from langchain-community instead:
`from langchain_community.llms import OpenAI`.

To install langchain-community run `pip install -U langchain-community`.
warnings.warn(
/Users/ambroquach/Desktop/jetai-project/llm/llm-env/lib/python3.12/site-packages/langchain_core/_api/deprecation.py:117: LangChainDeprecationWarning: The class 'langchain_community.llms.openai.OpenAI' was deprecated in langchain-community 0.0.10 and will be removed in 0.2.0. An updated version of the class exists in the langchain-openai package and should be used instead. To use it run `pip install -U langchain-openai` and import as `from langchain_openai import OpenAI`.
warn_deprecated(
INFO:     Started server process [85900]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:52613 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:52613 - "GET /openapi.json HTTP/1.1" 200 OK

```

## Setting up Frontend

28. Navigate to the **/frontend** directory and run **npm install** then **npm run dev** to start the NextJS app.
29. Go to <http://localhost:3000> to start using the Jet AI Comparison Tool

The screenshot shows a web application interface for comparing charter jets. At the top, there's a navigation bar with icons and a search bar. Below it is a table titled "Top 10 Charter Jets" with columns for SELECT, ID, NAME, WINGSPAN, NUMBER OF ENGINES, and MANUFACTURING YEAR. The table lists ten jets, each with a checkbox in the first column. The jets listed are: Gulfstream G650, Bombardier Global 7500, Cessna Citation Longitude, Embraer Phenom 300, Dassault Falcon 7X, Bombardier Challenger 350, Gulfstream G280, HondaJet Elite, Pilatus PC-24, and Learjet 75 Liberty. The last two rows (6 and 7) have checkboxes in the first column, while others have checkboxes in the second column. Below the table is a section titled "Ask OpenAI to Compare Selected Jets By:" with a dropdown menu set to "Top Speed (mph)" and a "Compare Selected Jets" button. A smaller table below shows the top speeds for the selected jets: Gulfstream G650 (704 mph), Dassault Falcon 7X (590 mph), Cessna Citation Longitude (548 mph), Embraer Phenom 300 (453 mph), and Bombardier Global 7500 (104 mph).

SELECT	ID	NAME	WINGSPAN	NUMBER OF ENGINES	MANUFACTURING YEAR
<input checked="" type="checkbox"/>	1	Gulfstream G650	99.7	2	2012
<input checked="" type="checkbox"/>	2	Bombardier Global 7500	104	2	2018
<input checked="" type="checkbox"/>	3	Cessna Citation Longitude	68.5	2	2017
<input checked="" type="checkbox"/>	4	Embraer Phenom 300	53.2	2	2009
<input checked="" type="checkbox"/>	5	Dassault Falcon 7X	86	3	2005
<input type="checkbox"/>	6	Bombardier Challenger 350	69	2	2014
<input type="checkbox"/>	7	Gulfstream G280	63	2	2011
<input type="checkbox"/>	8	HondaJet Elite	39.8	2	2018
<input type="checkbox"/>	9	Pilatus PC-24	55.8	2	2017
<input type="checkbox"/>	10	Learjet 75 Liberty	50.9	2	2020

**Ask OpenAI to Compare Selected Jets By:**

Top Speed (mph)

RANK	NAME	VALUE
1	Gulfstream G650	704
2	Dassault Falcon 7X	590
3	Cessna Citation Longitude	548
4	Embraer Phenom 300	453
5	Bombardier Global 7500	104

~Thank you for your time and reviewing this project. Feel free to reach out to me if you have any question or there's any hiccup in the process of getting it to run.~

## Other related screenshots/examples:

The screenshot shows the Hasura Data Manager interface. The top navigation bar includes links for API, DATA, ACTIONS, REMOTE SCHEMAS, EVENTS, Secure your endpoint, ENTERPRISE, SETTINGS, HELP, and user profile.

**Data Manager** section:

- Databases (1): jetdb (selected), public
- Search tables in public...
- Export data
- Sort: No sort conditions present.
- Add
- 100 rows
- Prev, Next buttons

		id	name	wingspan	numberOfEngines	manufacturingYear	createdAt	updatedAt
1	Gulfstream G650	99.7	2	2012	2024-03-15T14:12:48.526	2024-03-15T14:12:48.526		
2	Bombardier Global 7500	104	2	2018	2024-03-15T14:12:48.526	2024-03-15T14:12:48.526		
3	Cessna Citation Longitude	68.5	2	2017	2024-03-15T14:12:48.526	2024-03-15T14:12:48.526		
4	Embraer Phenom 300	53.2	2	2009	2024-03-15T14:12:48.526	2024-03-15T14:12:48.526		
5	Dassault Falcon 7X	86	3	2005	2024-03-15T14:12:48.526	2024-03-15T14:12:48.526		
6	Bombardier Challenger 350	69	2	2014	2024-03-15T14:12:48.526	2024-03-15T14:12:48.526		
7	Gulfstream G280	63	2	2011	2024-03-15T14:12:48.526	2024-03-15T14:12:48.526		
8	HondaJet Elite	39.8	2	2018	2024-03-15T14:12:48.526	2024-03-15T14:12:48.526		
9	Pilatus PC-24	55.8	2	2017	2024-03-15T14:12:48.526	2024-03-15T14:12:48.526		
10	Learjet 75 Liberty	50.9	2	2020	2024-03-15T14:12:48.526	2024-03-15T14:12:48.526		

**SQL**, **Native Queries**, **Model Summary** sections are also visible.

**Browser** section:

- Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (2), hdb\_catalog, public, Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables (2), Jet, \_prisma\_migrations, Trigger Functions, Types, Views, Subscriptions, Login/Group Roles, Tablespaces.
- Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Processes, postres/post..., public...\_prisma..., public.Jet/post..., scratch pad.
- Query History: SELECT \* FROM public."Jet" ORDER BY id ASC
- Scratch Pad

**Data Output** section:

id	name	wingspan	numberOfEngines	manufacturingYear	createdAt	updatedAt
1	Gulfstream G650	99.7	2	2012	2024-03-15 14:12:48.526	2024-03-15 14:12:48.526
2	Bombardier Global 7500	104	2	2018	2024-03-15 14:12:48.526	2024-03-15 14:12:48.526
3	Cessna Citation Longitude	68.5	2	2017	2024-03-15 14:12:48.526	2024-03-15 14:12:48.526
4	Embraer Phenom 300	53.2	2	2009	2024-03-15 14:12:48.526	2024-03-15 14:12:48.526
5	Dassault Falcon 7X	86	3	2005	2024-03-15 14:12:48.526	2024-03-15 14:12:48.526
6	Bombardier Challenger 350	69	2	2014	2024-03-15 14:12:48.526	2024-03-15 14:12:48.526
7	Gulfstream G280	63	2	2011	2024-03-15 14:12:48.526	2024-03-15 14:12:48.526
8	HondaJet Elite	39.8	2	2018	2024-03-15 14:12:48.526	2024-03-15 14:12:48.526

Total rows: 10 of 10 | Query complete 00:00:00.531 | Ln 1, Col 1

localhost:8080/console/api/api-explorer

**HASURA v2.37.0**

**API** DATA ACTIONS REMOTE SCHEMAS EVENTS **Secure your endpoint** **ENTERPRISE** SETTINGS HELP SECURITY

GraphQL REST Schema Registry

Allow List Security

**GraphQL Endpoint**

POST http://localhost:8080/v1/graphql

**Request Headers**

ENABLE	KEY	VALUE
<input checked="" type="checkbox"/>	content-type	application/json
	Enter Key	Enter Value

**Explorer**

query MyQuery {  
 Jet {  
 id  
 name  
 wingspan  
 numberofEngines  
 manufacturingYear  
 createdAt  
 updatedAt  
 }  
 Jet {  
 id  
 name  
 wingspan  
 numberofEngines  
 manufacturingYear  
 createdAt  
 updatedAt  
 }  
}

GraphIQL

Prettify History Explorer Cache Code Exporter REST Derive action Analyze Docs

API Explorer | Hasura Apollo Server Create Next App FastAPI - Swagger UI localhost:8080/api/rest/jets

localhost:8080/api/rest/jets

```
["Jet": [{"id": 1, "name": "Gulfstream G650", "wingspan": 99.7, "numberofEngines": 2, "manufacturingYear": 2012, "createdAt": "2024-03-15T14:12:48.526", "updatedAt": "2024-03-15T14:12:48.526"}, {"id": 2, "name": "Bombardier Global 7500", "wingspan": 104, "numberofEngines": 2, "manufacturingYear": 2018, "createdAt": "2024-03-15T14:12:48.526", "updatedAt": "2024-03-15T14:12:48.526"}, {"id": 3, "name": "Cessna Citation Longitude", "wingspan": 68.5, "numberofEngines": 2, "manufacturingYear": 2017, "createdAt": "2024-03-15T14:12:48.526", "updatedAt": "2024-03-15T14:12:48.526"}, {"id": 4, "name": "Embraer Phenom 300", "wingspan": 53.2, "numberofEngines": 2, "manufacturingYear": 2009, "createdAt": "2024-03-15T14:12:48.526", "updatedAt": "2024-03-15T14:12:48.526"}, {"id": 5, "name": "Dassault Falcon 7X", "wingspan": 86, "numberofEngines": 3, "manufacturingYear": 2005, "createdAt": "2024-03-15T14:12:48.526", "updatedAt": "2024-03-15T14:12:48.526"}, {"id": 6, "name": "Bombardier Challenger 350", "wingspan": 69, "numberofEngines": 2, "manufacturingYear": 2014, "createdAt": "2024-03-15T14:12:48.526", "updatedAt": "2024-03-15T14:12:48.526"}, {"id": 7, "name": "Gulfstream G280", "wingspan": 63, "numberofEngines": 2, "manufacturingYear": 2011, "createdAt": "2024-03-15T14:12:48.526", "updatedAt": "2024-03-15T14:12:48.526"}, {"id": 8, "name": "HondaJet Elite", "wingspan": 39.8, "numberofEngines": 2, "manufacturingYear": 2018, "createdAt": "2024-03-15T14:12:48.526", "updatedAt": "2024-03-15T14:12:48.526"}, {"id": 9, "name": "Pilatus PC-24", "wingspan": 55.8, "numberofEngines": 2, "manufacturingYear": 2017, "createdAt": "2024-03-15T14:12:48.526", "updatedAt": "2024-03-15T14:12:48.526"}, {"id": 10, "name": "LEARjet 75 Liberty", "wingspan": 50.9, "numberofEngines": 2, "manufacturingYear": 2020, "createdAt": "2024-03-15T14:12:48.526", "updatedAt": "2024-03-15T14:12:48.526"}]]
```

localhost:5000/graphql

SANDBOX http://localhost:5000/graphql Publish

Documentation Root > Query > GetJetById

GetJetById: Jet

Arguments id: ID!

Fields id: ID!, name: String!, wingspan: Float!, numberofEngines: Int!, manufacturingYear: Int!, createdAt: DateTime!, updatedAt: DateTime!

Operation JetsQuery

```
query JetsQuery($getJetByIdId: ID!) {
  GetJetById(id: $getJetByIdId) {
    id
    name
    wingspan
    numberofEngines
    manufacturingYear
    createdAt
    updatedAt
  }
  GetAllJets {
    id
    name
    wingspan
    numberofEngines
  }
}
```

Variables

Headers Pre-Operation Script Post-Operation Script

```
JSON
1 [{}]
2   "getJetByIdId": "1"
3 ]
```

Response STATUS 200 | 116ms | 0B

```
data: {
  "GetJetById": {
    "id": "1",
    "name": "Gulfstream G650",
    "wingspan": 99.7,
    "numberofEngines": 2,
    "manufacturingYear": 2012,
    "createdAt": "2024-03-15T14:12:48.526",
    "updatedAt": "2024-03-15T14:12:48.526"
  },
  "GetAllJets": [
    {
      "id": "1",
      "name": "Gulfstream G650",
      "wingspan": 99.7,
      "numberofEngines": 2,
      "manufacturingYear": 2012,
      "createdAt": "2024-03-15T14:12:48.526",
      "updatedAt": "2024-03-15T14:12:48.526"
    },
    {
      "id": "2",
      "name": "Bombardier Global 7500",
      "wingspan": 104,
      "numberofEngines": 2,
      "manufacturingYear": 2018,
      "createdAt": "2024-03-15T14:12:48.526",
      "updatedAt": "2024-03-15T14:12:48.526"
    }
  ]
},
```

localhost:5000/graphql

SANDBOX http://localhost:5000/gra... Publish Log in

Documentation

Operation InsertJet

```
1 mutation InsertJet($name: String!, $wingspan: Float!, $numberOfEngines: Int!, $manufacturingYear: Int!) {  
2   InsertJet(name: $name, wingspan: $wingspan,  
3   numberOfEngines: $numberOfEngines, manufacturingYear:  
4   $manufacturingYear) {  
5     id  
6     name  
7     wingspan  
8     numberOfEngines  
9     manufacturingYear  
10    createdAt  
11    updatedAt  
12  }  
13}
```

Response STATUS 200 | 3.04s | 214B

```
{"data": {  
  "InsertJet": {  
    "id": "11",  
    "name": "Honda Jet Supercharge",  
    "wingspan": 154,  
    "numberOfEngines": 6,  
    "manufacturingYear": 2030,  
    "createdAt": "2024-03-13T06:52:29.123+00:00",  
    "updatedAt": "2024-03-13T06:52:29.123+00:00"  
  }  
}}
```

Variables Headers Pre-Operation Script Post-Operation Script JSON

```
1 [  
2   "name": "Honda Jet Supercharge",  
3   "wingspan": 154,  
4   "numberOfEngines": 6,  
5   "manufacturingYear": 2030  
6 ]
```

+ Add files