Microsoft

public preview

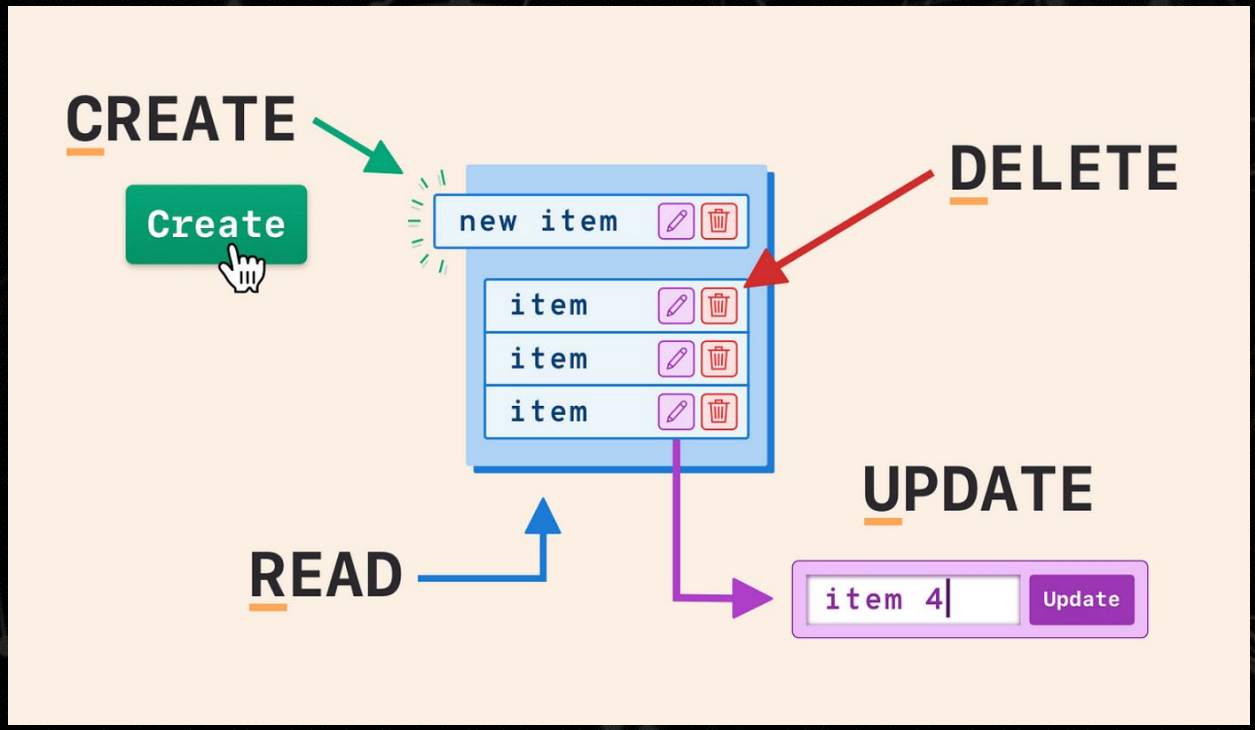
Data API builder

`https://aka.ms/dab`

Varun Dhawan | iVarund

Product Manager | Azure PostgreSQL

API 101 - Curd Vs CRUD



😋

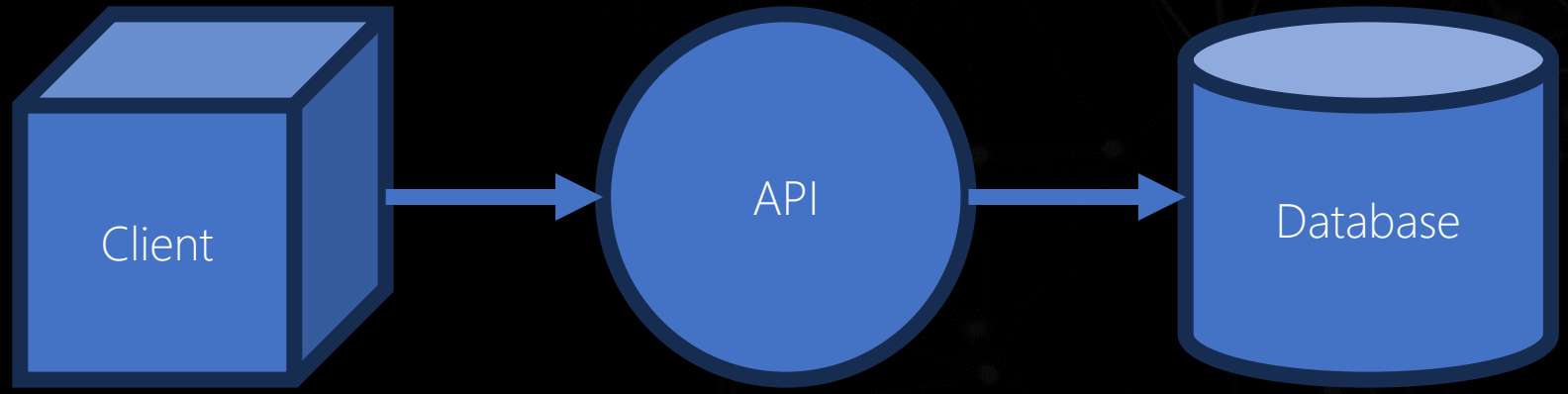
💻

About Me

- Product Manager at Microsoft
- Over 20 years experience working on relational systems – PostgreSQL, Oracle, SQL Server
- Previously worked as DevOps engineer at Target and DBA at McKinsey&Company
- My fav things – Hiking and Blogging > data-nerd.blog

Standard Data API

public preview



Engines not more code

public preview

DEMO

Data API Builder

Ready to get start

```
C:\dotnet tool install -g microsoft.dataapibuilder
```

Step 1. DAB INIT

```
>dab init
--database-type "postgresql"
--connection-string "server=<name>;database=dab;port=5432;user id=pg1user;"
--host-mode "Development"
```

Step 2. DAB ADD

```
-- Table 1. book
> dab add book --config "dab-config.json" --source books --permissions "anonymous:create,read,update,delete"
-- Table 2. authors
> dab add author --config "dab-config.json" --source authors --permissions "anonymous:create,read,update,dele
```

Step 3. DAB START

```
> Dab start
```

Built-in developer tooling

REST testing

https://localhost:5001/swagger

Graph QL testing

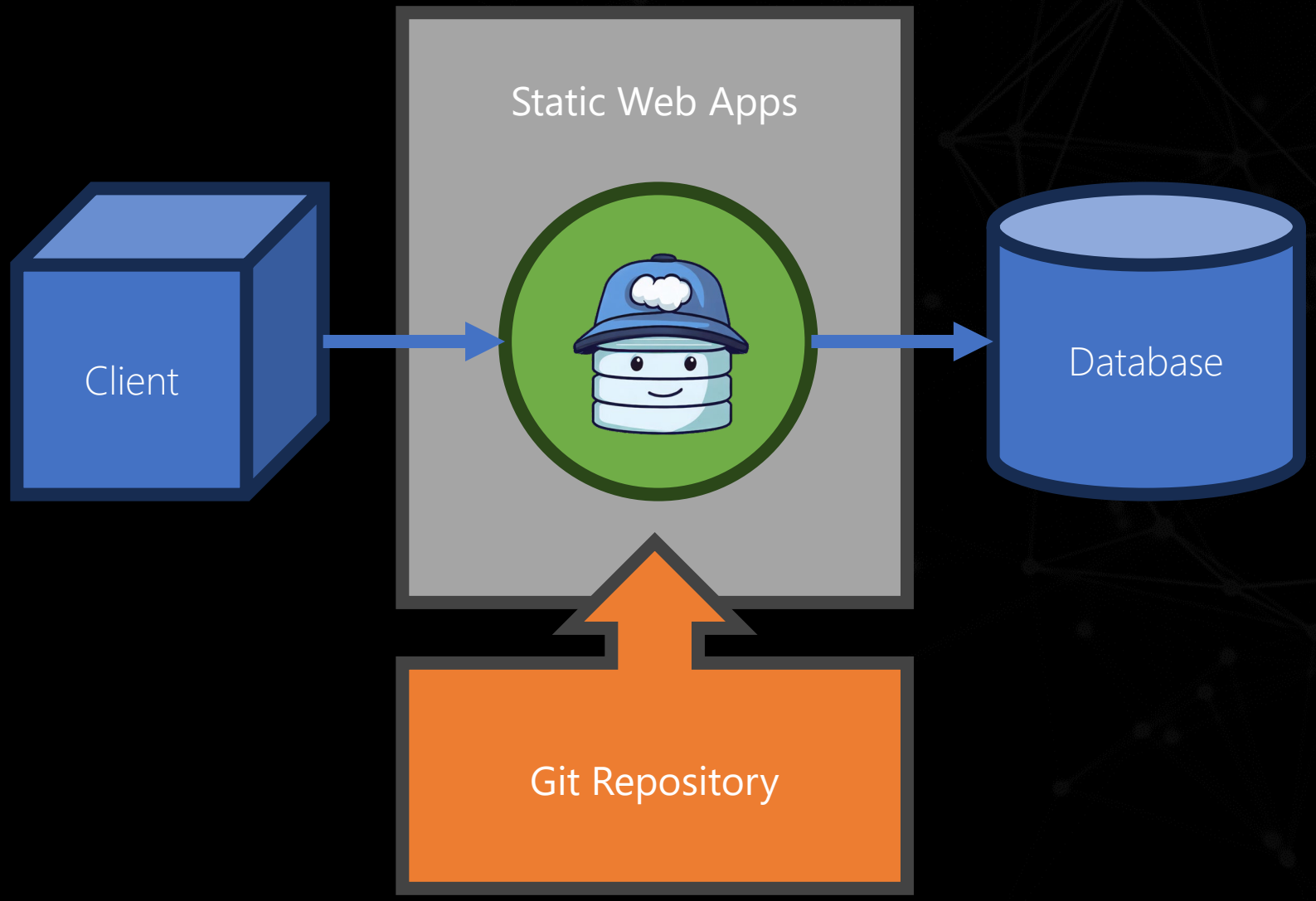
https://localhost:5001/graphql

Schema declaration

https://localhost:5001/api/openapi

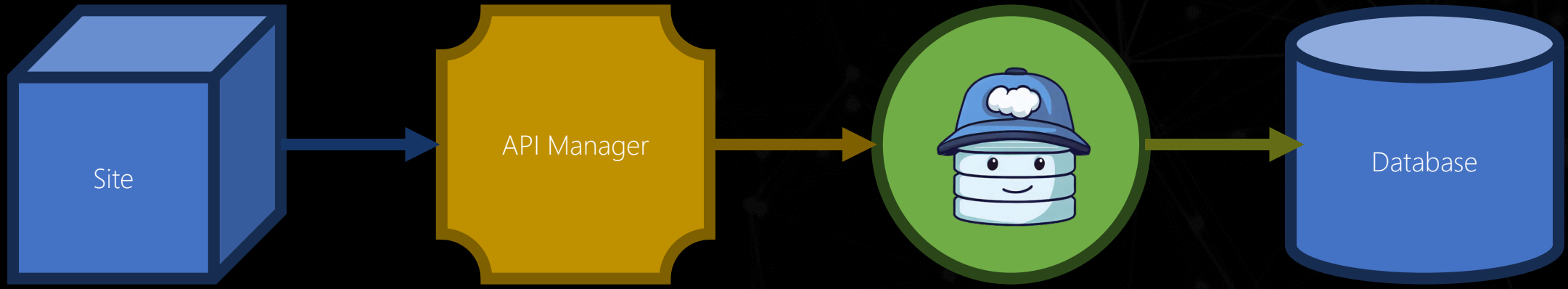
public preview

Easy to Start

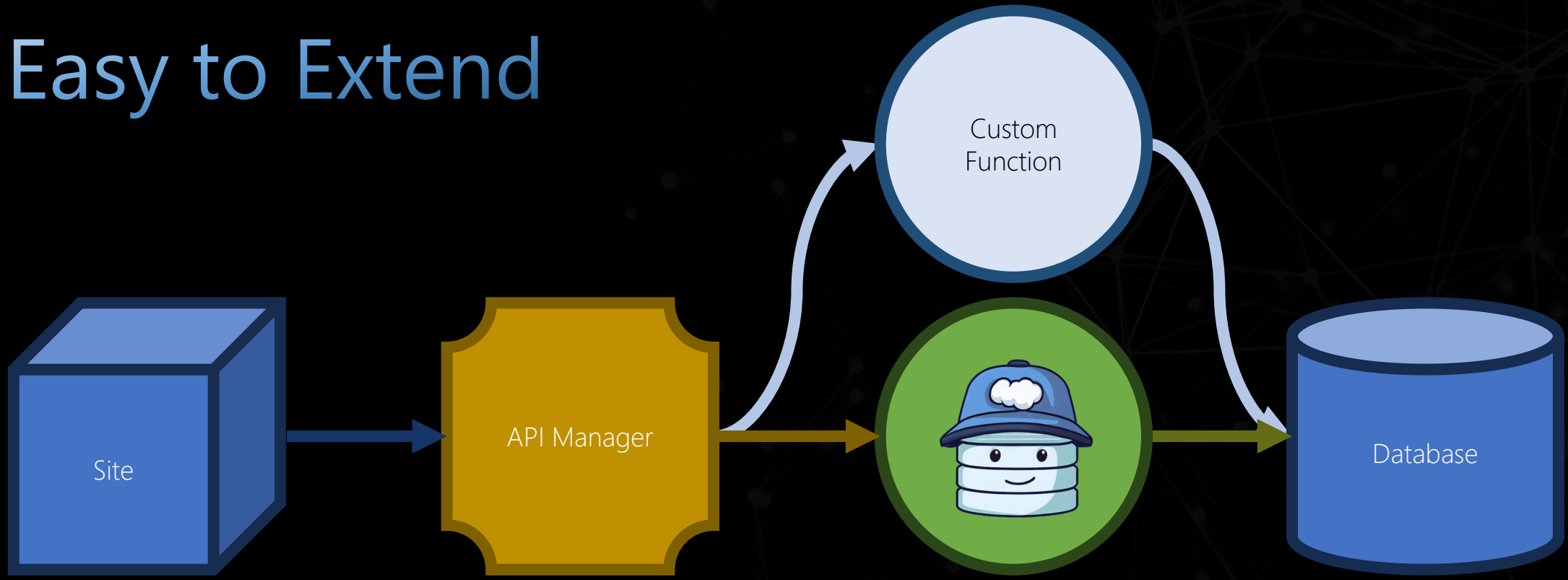


public preview

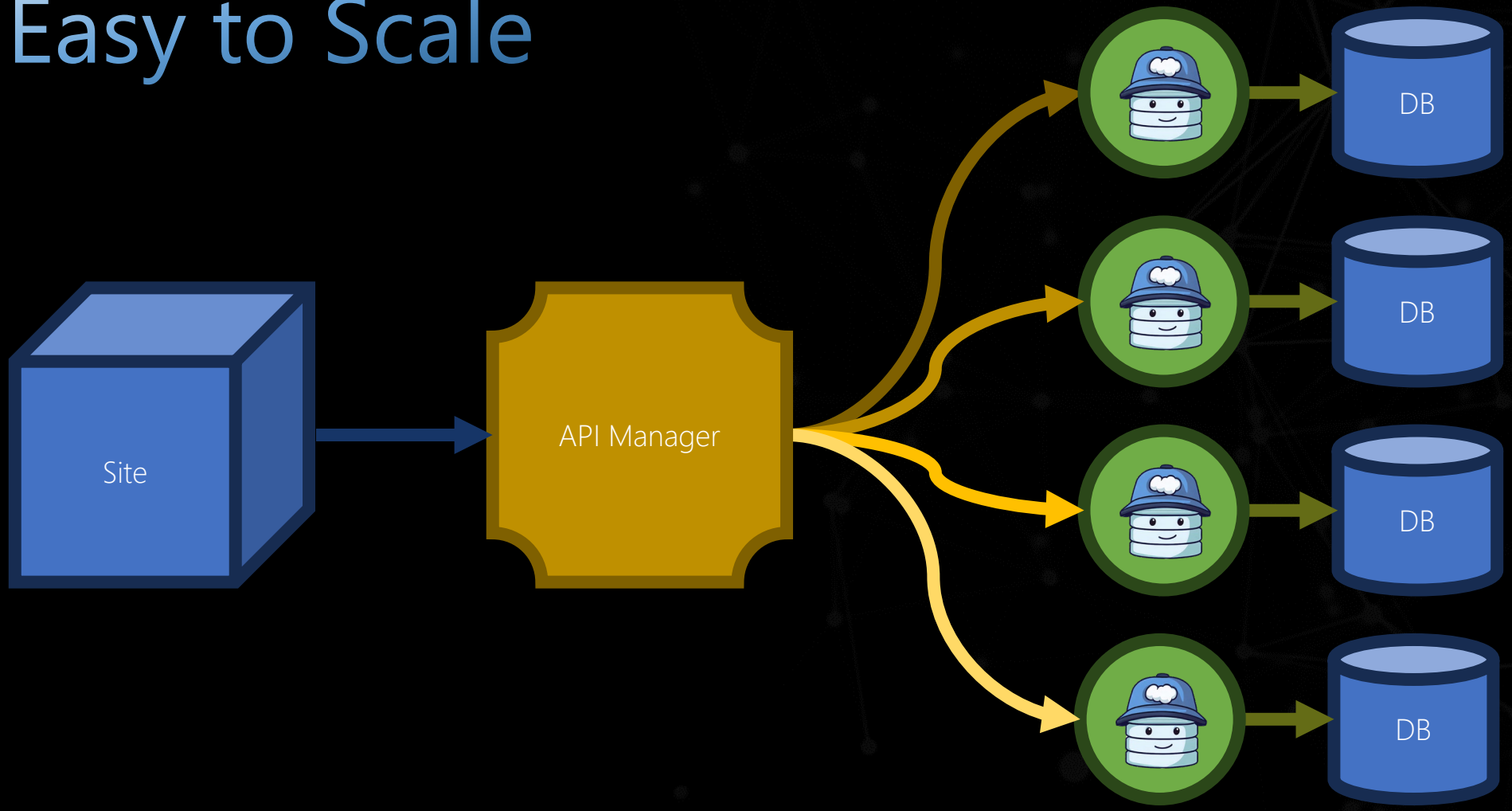
Easy to Manage



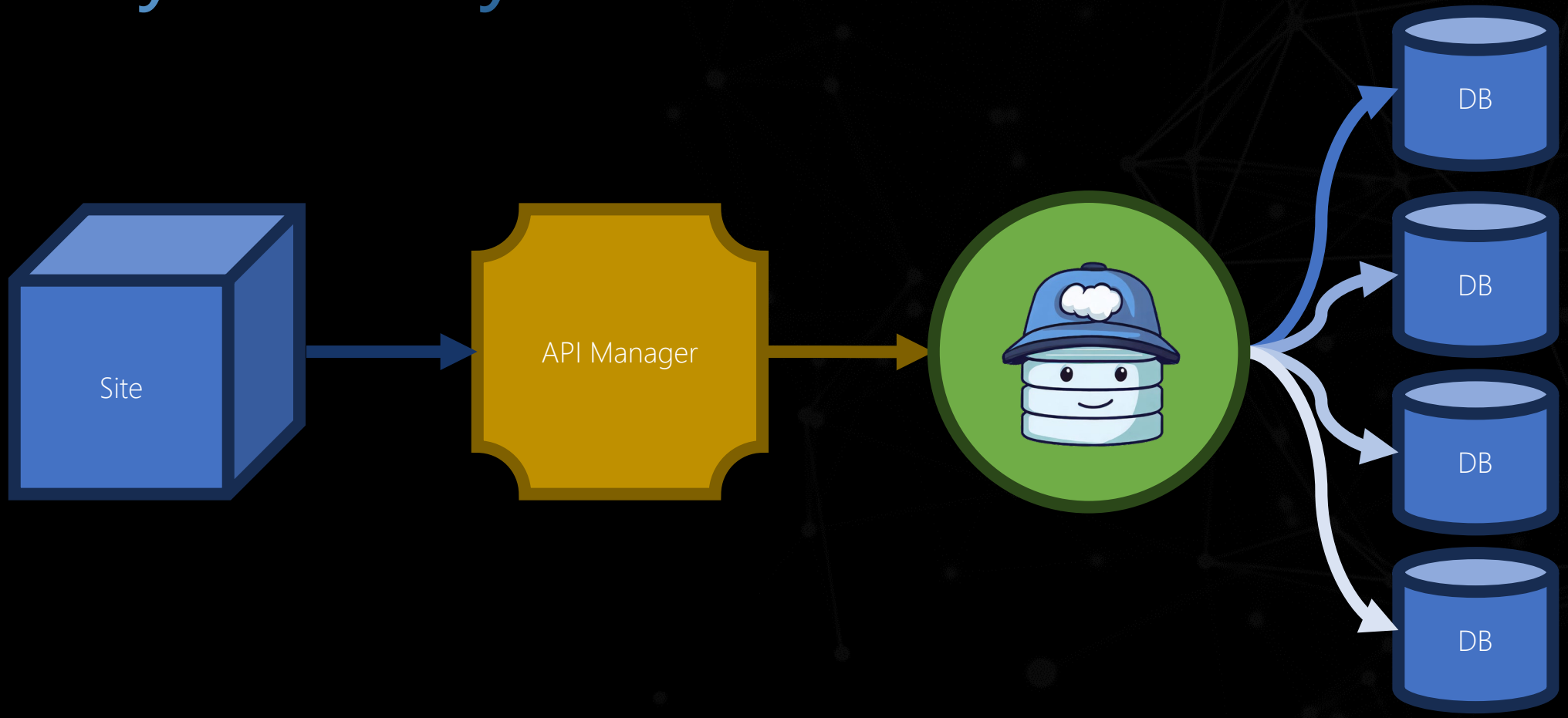
Easy to Extend



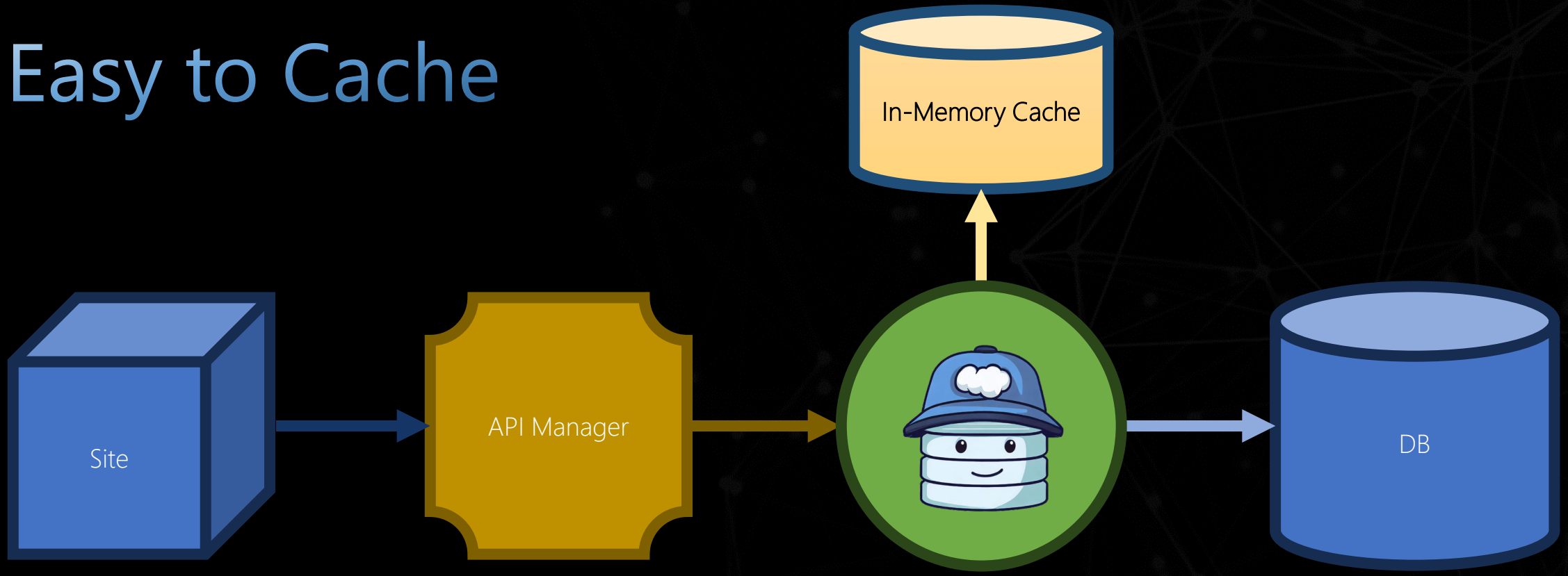
Easy to Scale



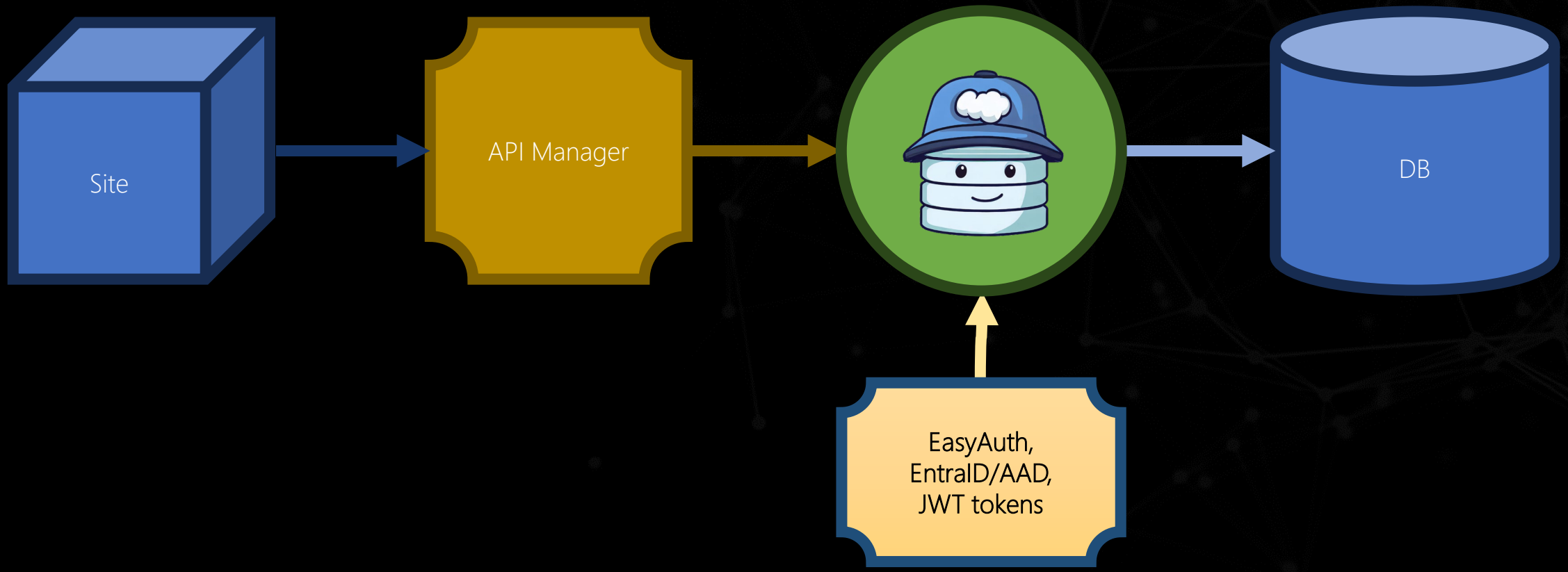
Easy to Unify



Easy to Cache

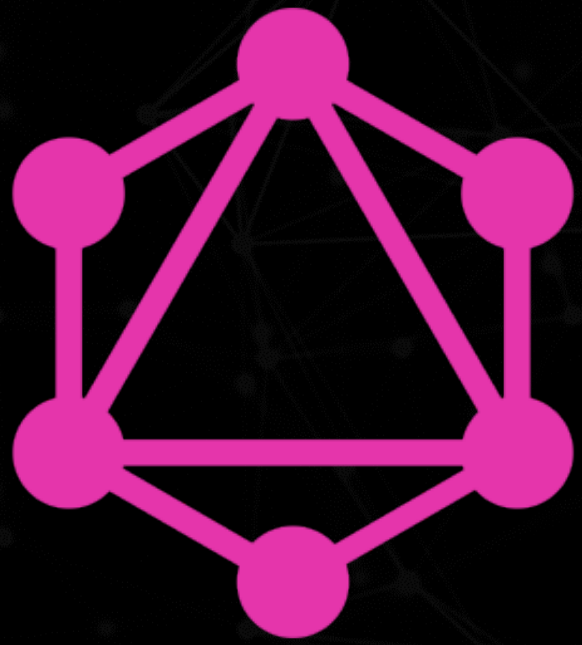


Easy to Secure



A *Modern* API

&

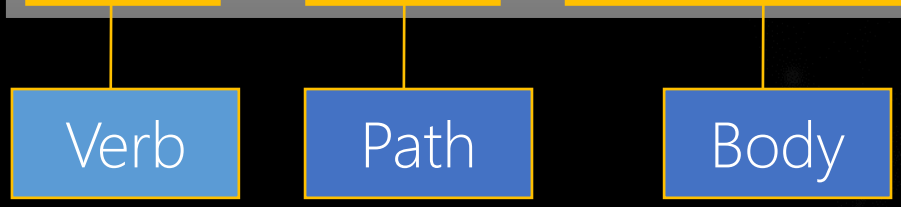


REST

GraphQL

REST -refresher-

```
POST /user {name: ‘Jerry’}
```




```
PUT (UPDATE)
GET (SELECT)
POST (INSERT)
DELETE (DELETE)
```

REST + ODATA -refresher-

```
GET /user?$select=Id,Name&$filter=Id eq 123
```

Verb

OData syntax

```
$select (SELECT)
$filter (WHERE)
$orderby (ORDER BY)
$first (TOP)
$after (OFFSET)
```

GraphQL -refresher-

```
POST /graphql { query / mutation }
```

Verb

Schema

Query

```
type User {
   name: String!,
   email: String,
   manager: String,
   reports: [Report]
}

type Report {
   name: String!,
   email: String,
}
```

```
query QueryName ($name: String!) {
   user (name: $name) {
      name,
      email,
      manager,
      reports {
         name,
         email
      }
   }
}
```

GraphQL -refresher-

```
POST /graphql { query / mutation }
```

Verb

Schema

Mutation

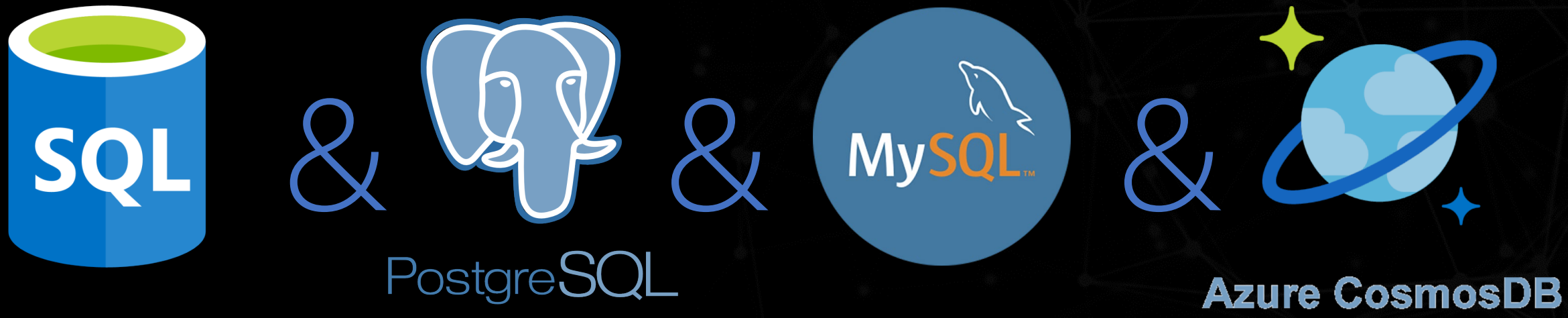
```
type Mutation {
   InsertUser(user: Input!): User
   UpdateUser(name: Input!): User
   DeleteUser (name: $name)
}

type Input {
   name: String!,
   email: String
}
```

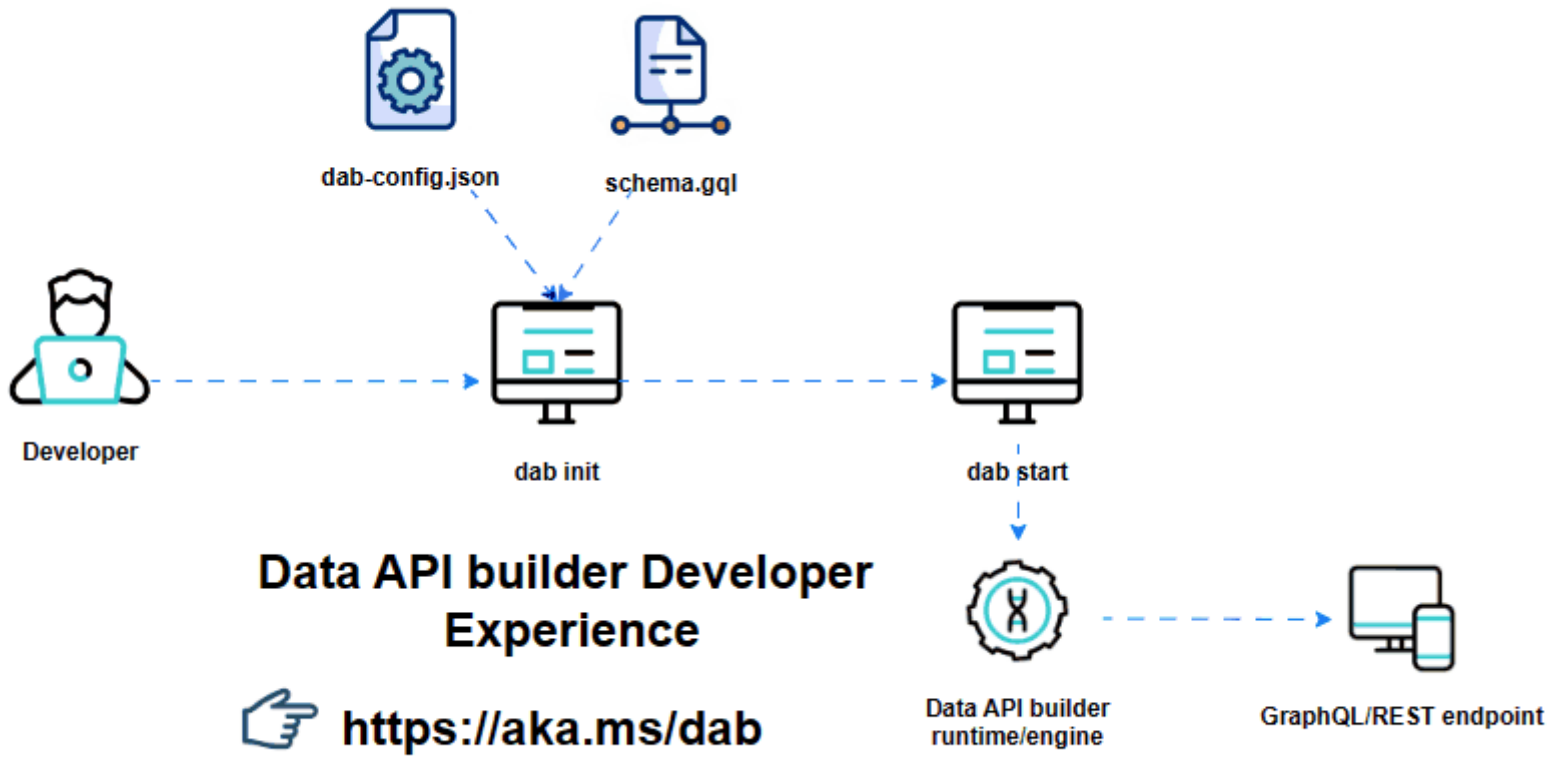
```
mutation MutName ($user: Input!) {
   insertUser (user: $user) {
      name,
      email
   }
}

{
   "name": "Jerry"
   "email": "email@domain.com"
}
```

All the above



* Plus Synapse support coming soon.


dab-config.json
schema.gql
Developer
dab init
dab start
Data API builder Developer Experience
https://aka.ms/dab
Data API builder runtime/engine
GraphQL/REST endpoint

public preview

Resources

DAB Repository: https://aka.ms/dab/repo

DAB Documentation: https://aka.ms/dab/docs

DAB Container Registry: https://aka.ms/dab/registry

DAB Samples

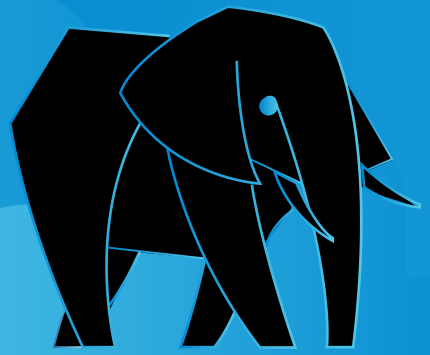
- https://aka.ms/dab/samples/one
- https://aka.ms/dab/samples/two
- https://aka.ms/dab/samples/three

Other Resources

- SQL Server Management Studio: https://aka.ms/ssms
- Azure Data Studio: https://aka.ms/azuredatastudio

Microsoft

Got 2 minutes?
We'd love your input
on how you use Postgres



Get your FREE socks
@ Microsoft booth

Connect with me

Varun Dhawan

Twitter – iVarund

LinkedIn - https://www.linkedin.com/in/varundhawan/

Blog - https://data-nerd.blog/