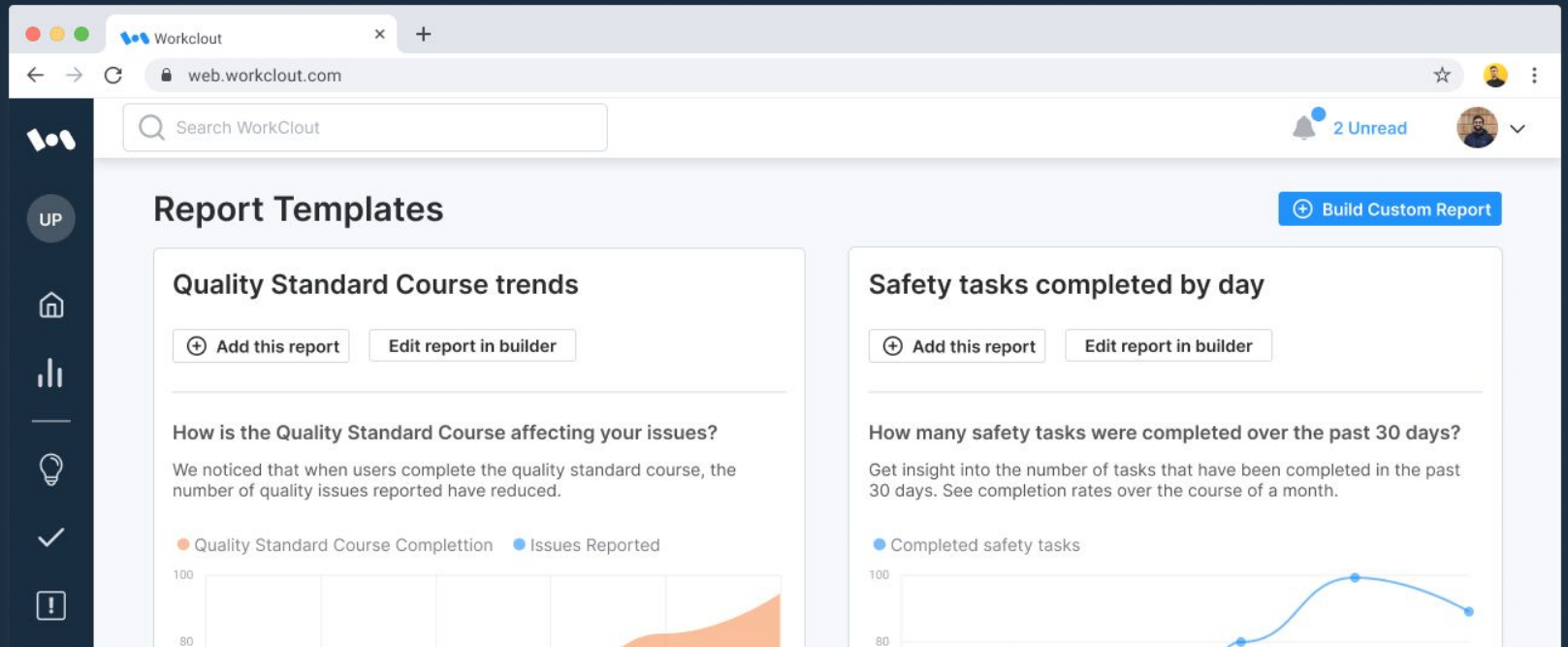
Building an enterprise multi-tenant SaaS product with Hasura

WorkClout

Enterprise Product for Frontline Workers and Continuous Improvement Managers

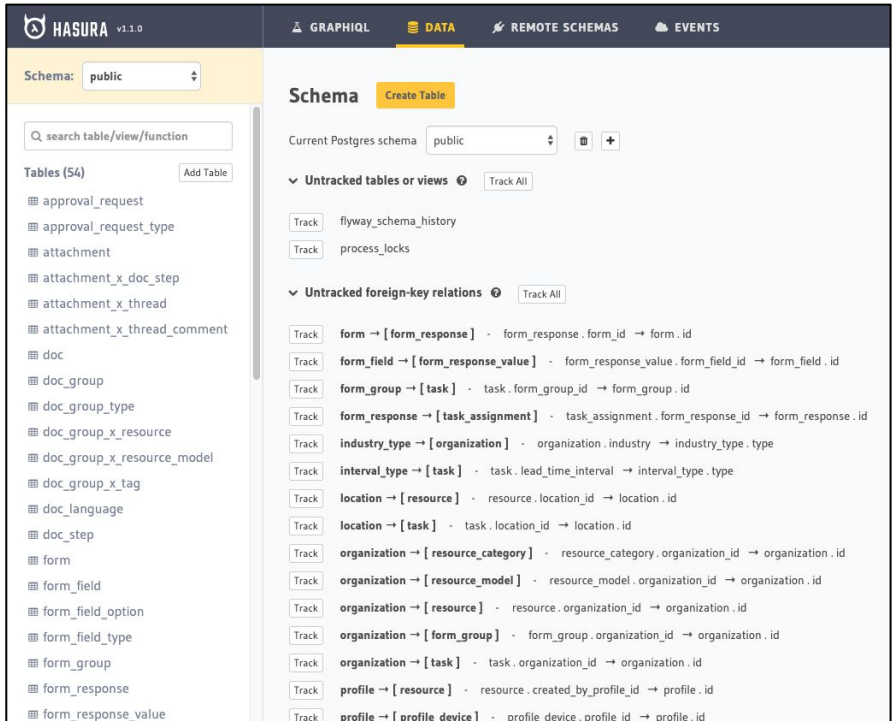


WorkClout

Enterprise Product for Frontline Workers and Continuous Improvement Managers

- Enterprise Users
- Real-time
- Multi-tenant
- SQL-based & Scalable

Our Journey Adopting Hasura



1. R&D + Implementation
2. Deployment
3. Maintenance

False Dichotomy

vs

This comparison primarily serviced the Frontend.

It obfuscated the benefits Hasura brings to the Backend.

HASURA

Flexibility to...

B.Y.O.B.T.

Bring your own backend team

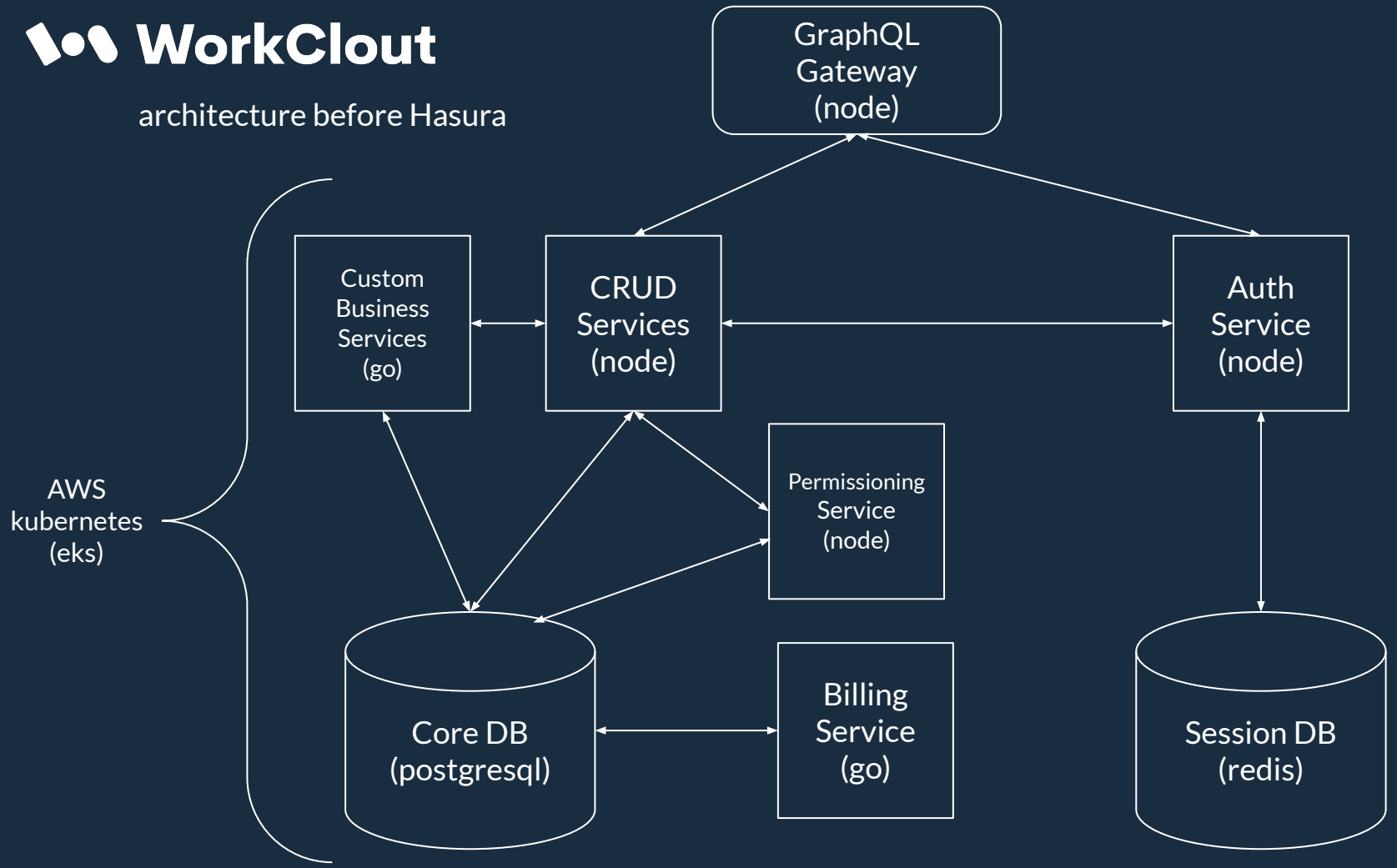
B.Y.O.B.S.

Bring your own backend stack

...or don't 😎

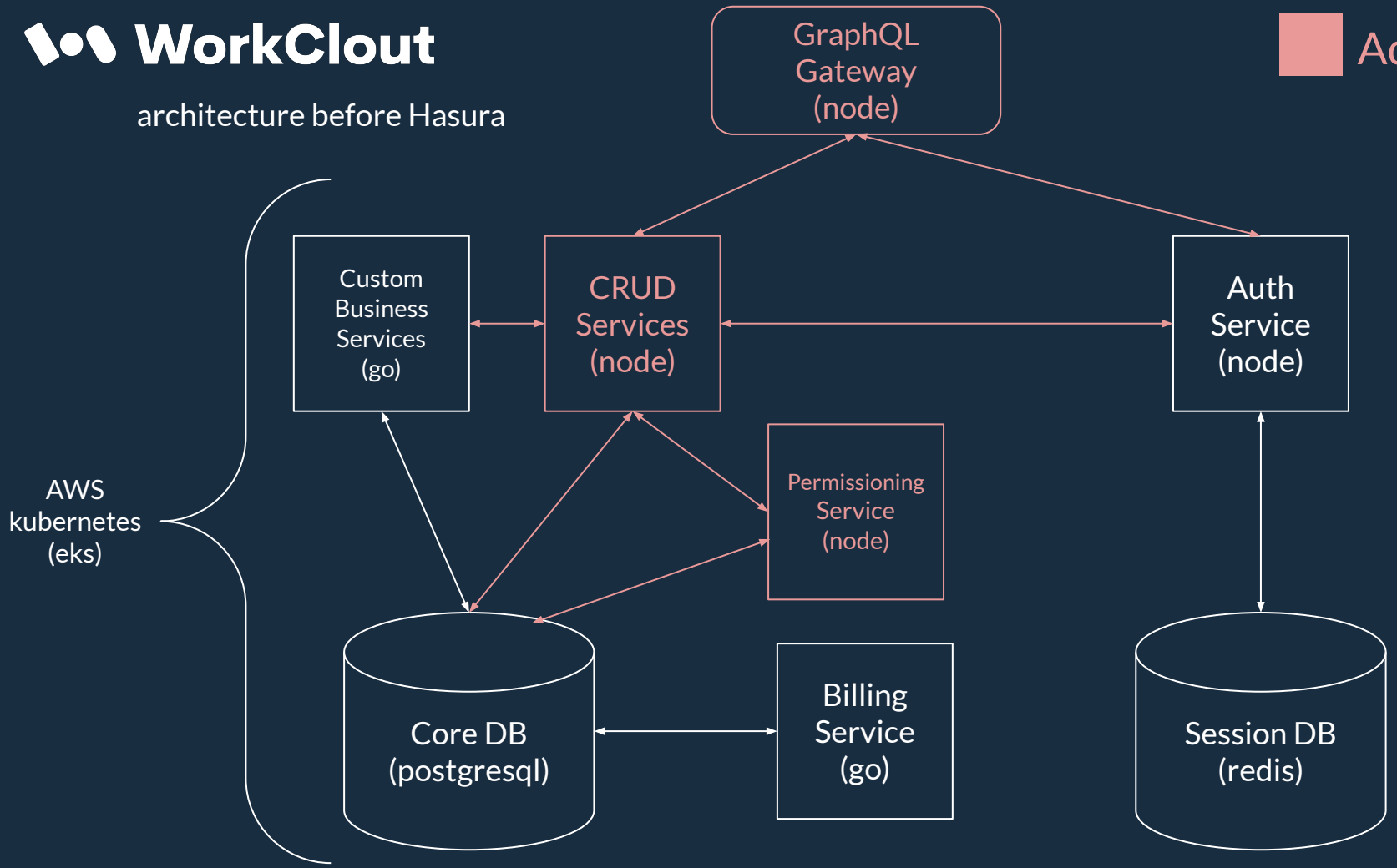
WorkClout

architecture before Hasura



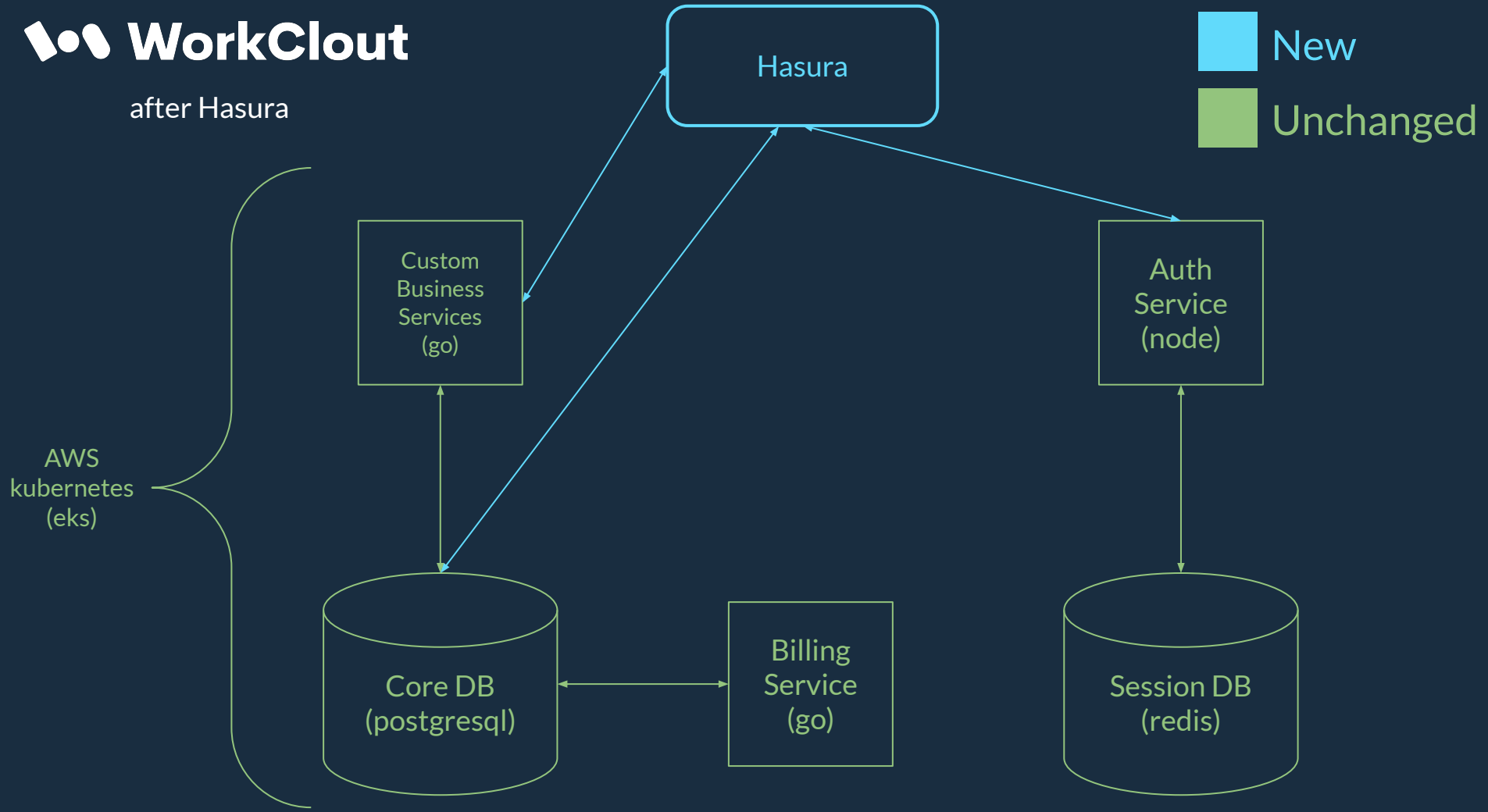
WorkClout

architecture before Hasura



WorkClout

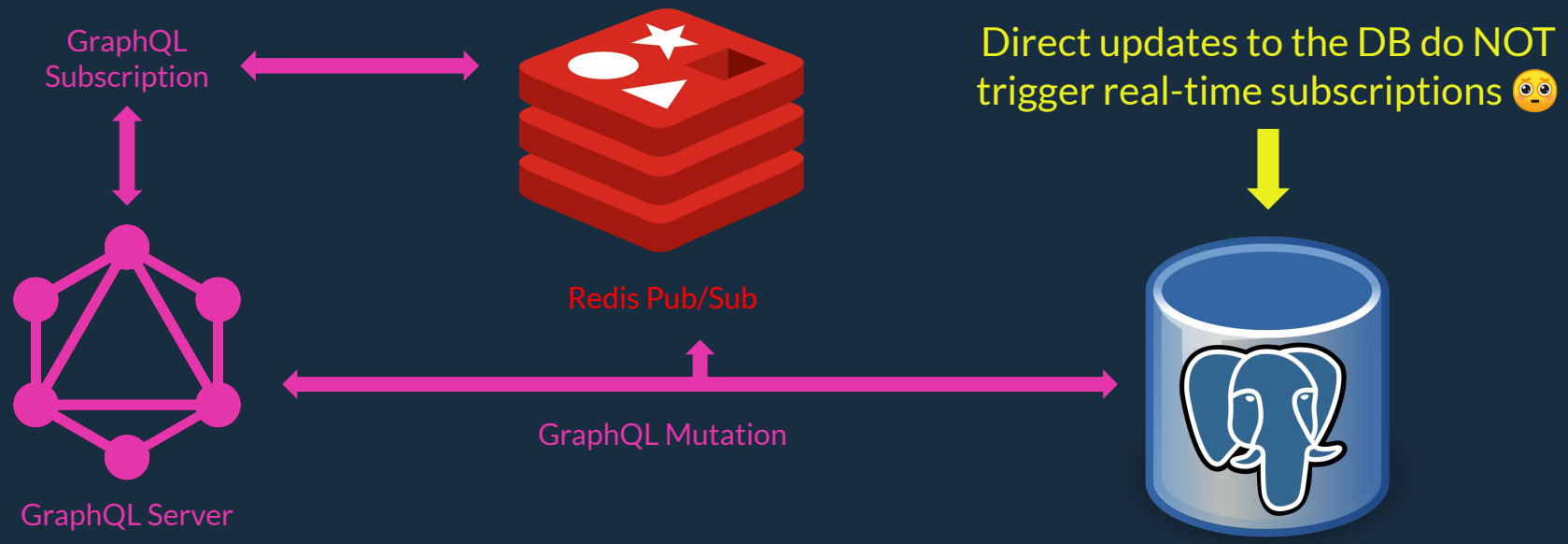
after Hasura



Just how real-time is real-time?

Just how real-time is real-time?

The old way:



Just how real-time is real-time?

The new way is much simpler:

Just how real-time is real-time?

```
hasurarox=# INSERT INTO assignment (name) VALUES ('Present at Hasura Con');
INSERT 0 1
```

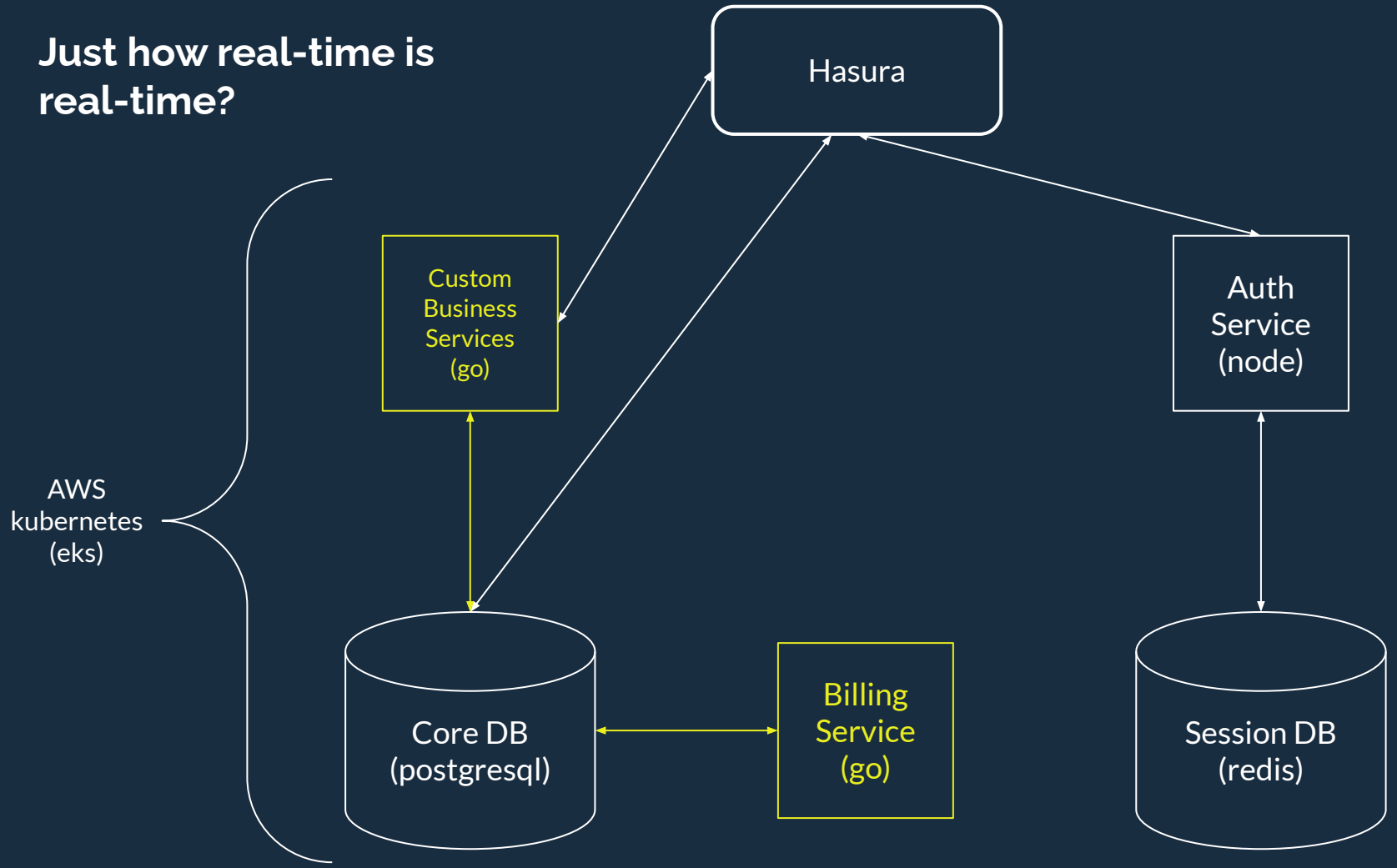
GraphiQL

Prettify | History | Copy | Explorer | Voyager | Analyze

```
subscription AllAssignments {
  assignment {
    id
    name
    created_at
  }
}
```

```
{
  "data": {
    "assignment": [
      {
        "id": "d0873a7a-c119-4f32-ae5a-83d0b3525172",
        "name": "Present at Hasura Con",
        "created_at": "2020-05-25T17:19:09.561132-07:00"
      }
    ]
  }
}
```

Just how real-time is real-time?



Choosing a GraphQL Client

Using Relay, we struggled with:

1. Relay Connection spec
2. Cache management + invalidation
3. Subscription management

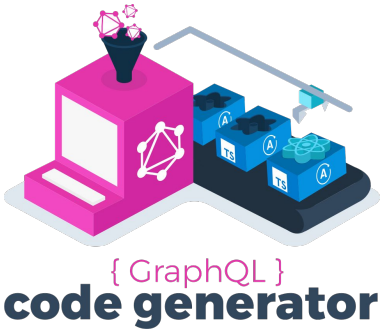
Relay

=

Hasura Team is working on Relay Support:
https://github.com/hasura/graphql-engine/issues/721

Choosing a GraphQL Client

+



=

Apollo Client, GraphQL Codegen, and use of React hooks

worked seamlessly with Hasura and Hasura subscriptions

Type-safe auto-generated React hooks for executing Hasura subscriptions

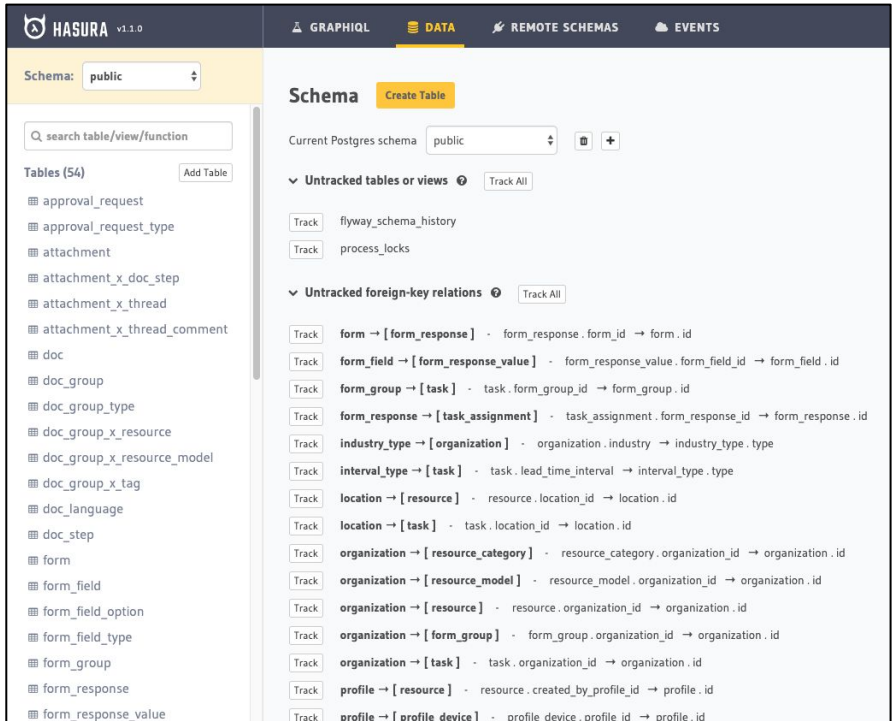
TS graphql.ts

```
1   import { gql } from "apollo-boost";
2
3   gql`
4     subscription TaskDetails($id: uuid!) {
5       task(where: { id: { _eq: $id } }) {
6         id
7         type
8         name
9         description
10        totalProfiles: profiles_aggregate {
11          aggregate {
12            count
13          }
14        }
15      }
16    }
17  `;
18
19
20
21
22
23
```

App.tsx

```
1   import React from 'react';
2   import { useTaskDetailsSubscription } from '../generated-graphql';
3   import Loader from '../Loader';
4   import Text from '../Text';
5
6   interface Props {
7     id: string;
8   }
9
10  export default ({ id }: Props) => {
    const { loading, data } = useTaskDetailsSubscription({
12    variables: {
13      id,
14    },
15  });
16
17  if (loading) {
18    return <Loader />;
19  }
20
21  return <Text>{data?.task[0].name}</Text>;
22  };
```

Our Journey Adopting Hasura

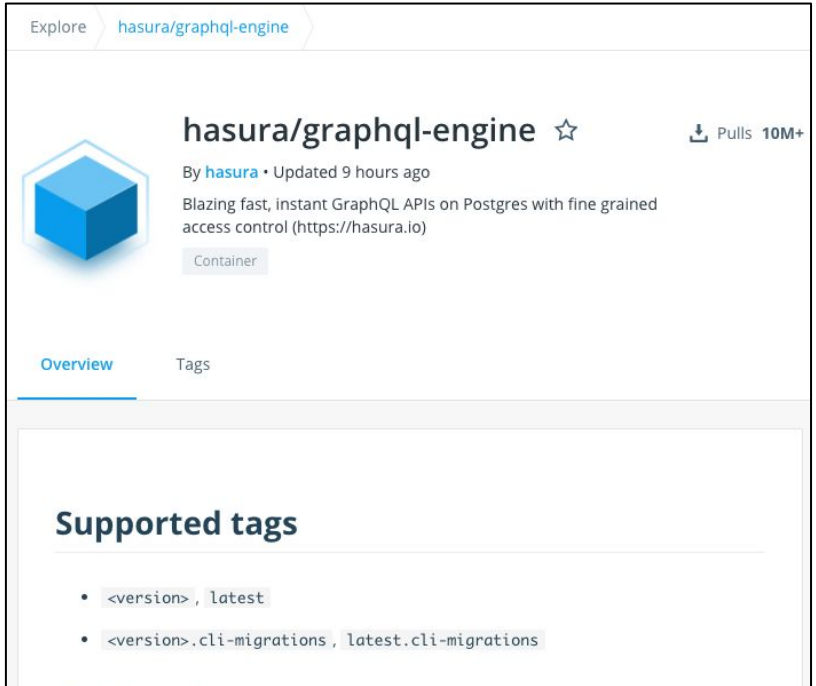


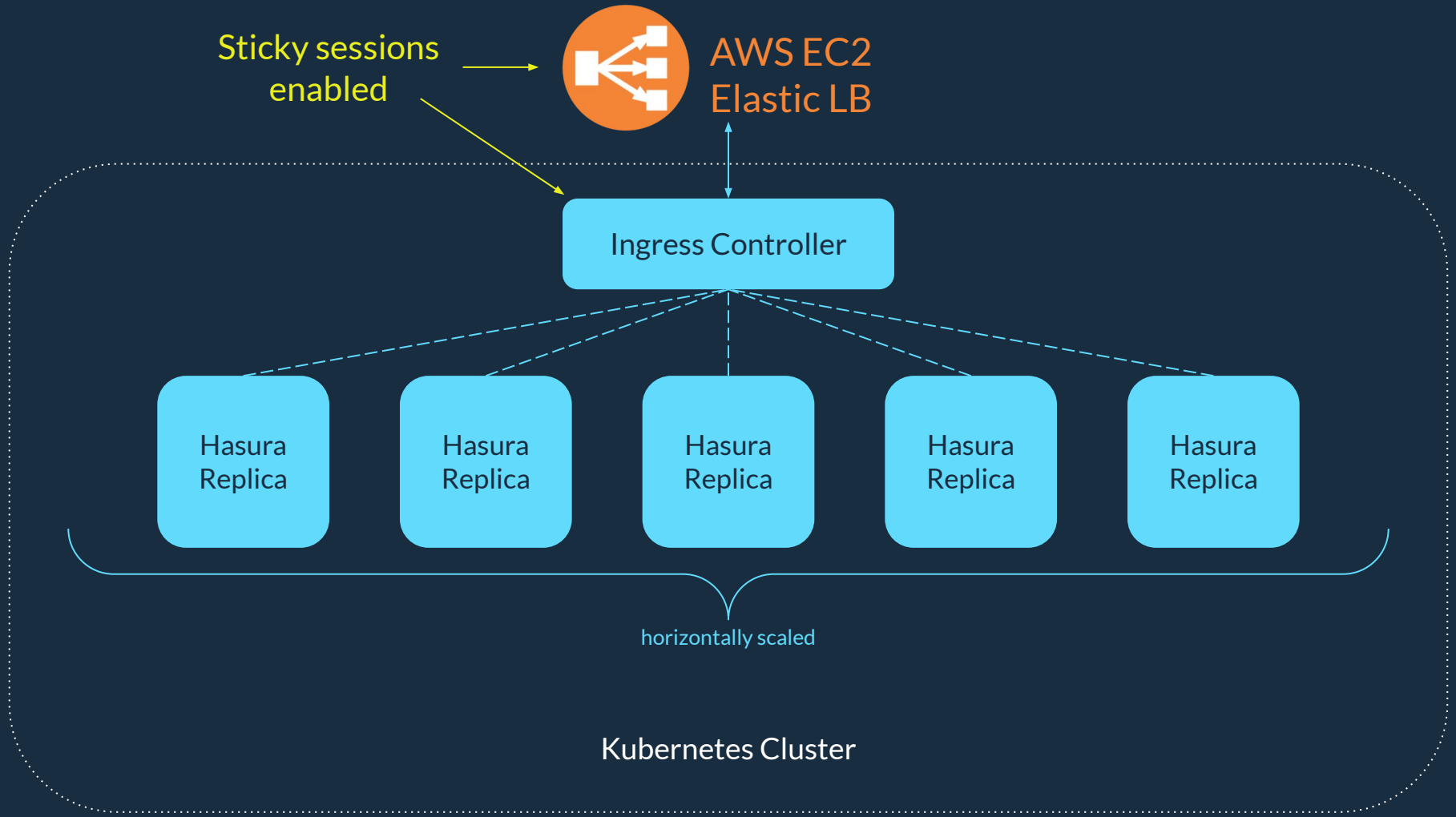
1. R&D + Implementation
2. Deployment
3. Maintenance

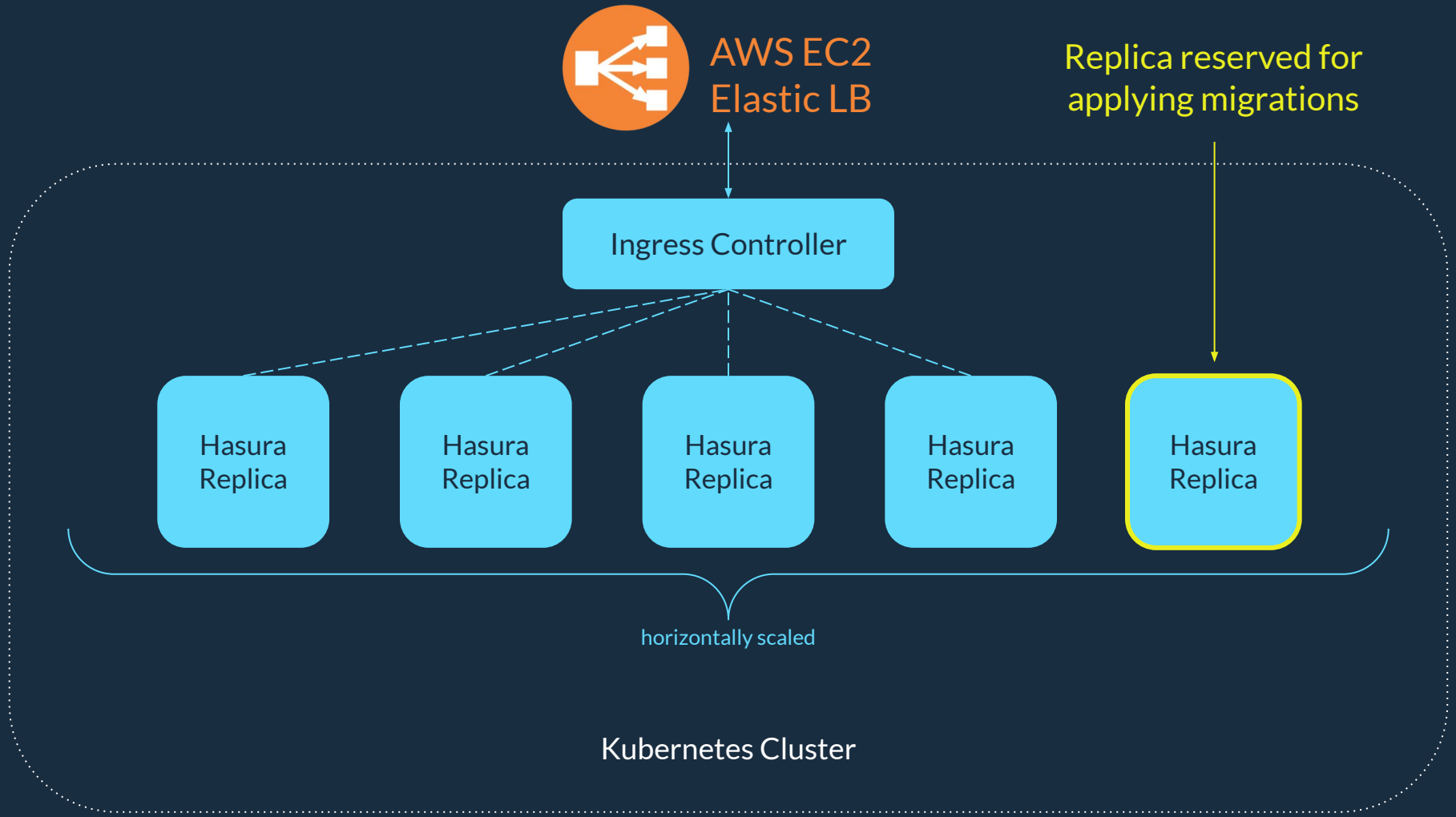
High Level Strategy

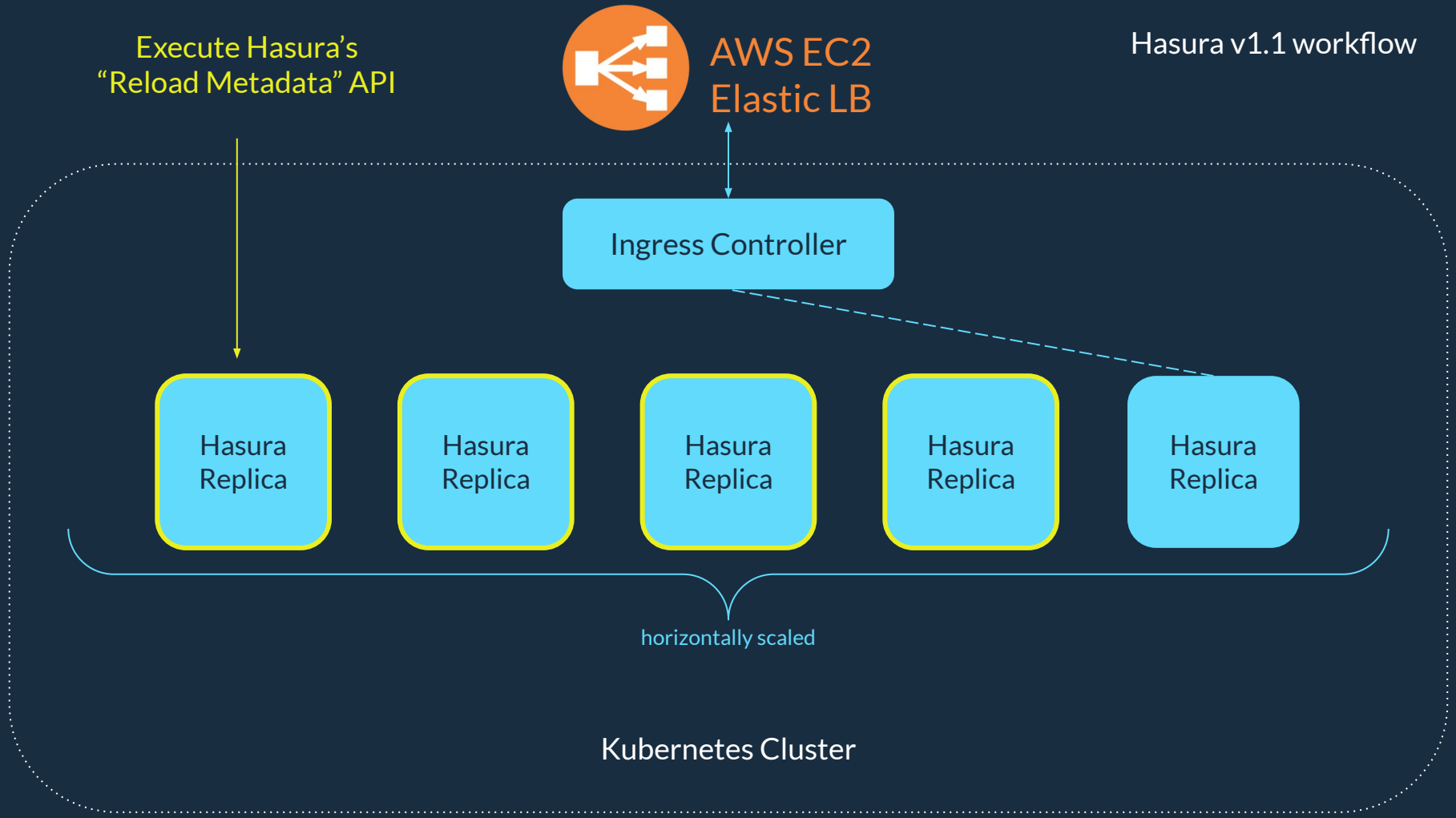
kubernetes

K8 deployment made easy thanks to...




Sticky sessions enabled
AWS EC2 Elastic LB
Ingress Controller
Hasura Replica
Hasura Replica
Hasura Replica
Hasura Replica
Hasura Replica
horizontally scaled
Kubernetes Cluster


AWS EC2
Elastic LB
Replica reserved for
applying migrations
Ingress Controller
Hasura
Replica
Hasura
Replica
Hasura
Replica
Hasura
Replica
Hasura
Replica
horizontally scaled
Kubernetes Cluster


Hasura v1.1 workflow
Execute Hasura's
"Reload Metadata" API
AWS EC2
Elastic LB
Ingress Controller
Hasura
Replica
Hasura
Replica
Hasura
Replica
Hasura
Replica
Hasura
Replica
horizontally scaled
Kubernetes Cluster

How about multi-tenancy?

Two Approaches to Multi-tenancy

On the DB level w/ “Data Hotelling”

id	organization_id	email	name
1	1	richardgirges@gmail.com	Richard
2	1	ariel@foo.com	Ariel
3	2	alastor@foo.com	Alastor
4	2	travis@foo.com	Travis
5	3	erica@foo.com	Erica

On the K8 Cluster level with Namespaces

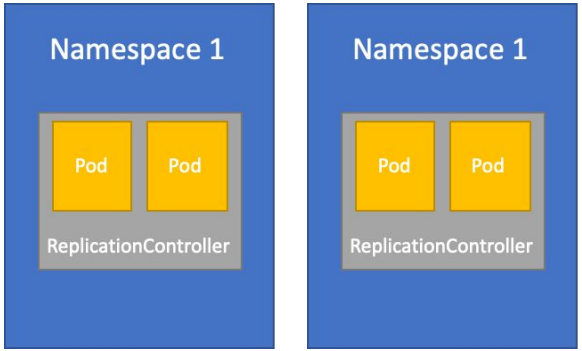
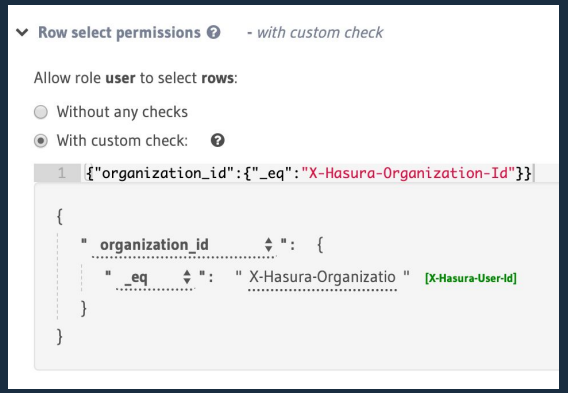


image source:
https://aws.amazon.com/blogs/containers/multi-tenant-design-considerations-for-amazon-eks-clusters/

We went with both.

Multi-tenancy

SMB customers share one DB. Hasura permissions makes it easy “hotel” their data



Shared database with Data Hotelling

Shared K8 Namespace

Large enterprises get a dedicated DB, Hasura instance, and K8 namespace

Dedicated DB

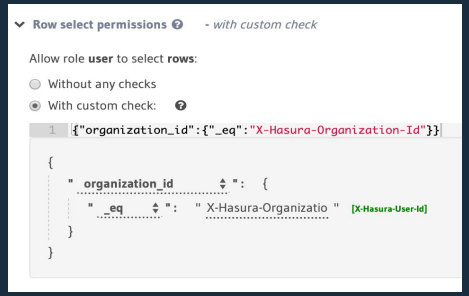
Enterprise Cust1 K8 Namespace

Dedicated DB

Enterprise Cust2 K8 Namespace

Multi-tenancy

One multi-tenant K8 cluster for all customers



Shared database with Data Hotelling

Shared K8 Namespace

Hasura Replicas

Dedicated DB

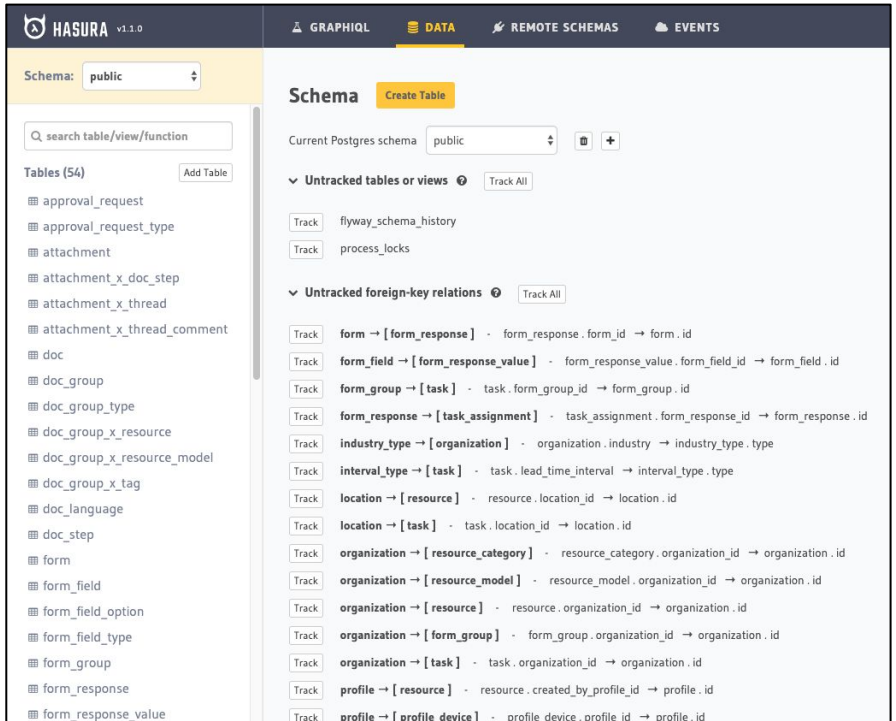
Enterprise Cust 1 K8 Namespace

Hasura Replicas

Dedicated DB

Enterprise Cust 2 K8 Namespace

Our Journey Adopting Hasura



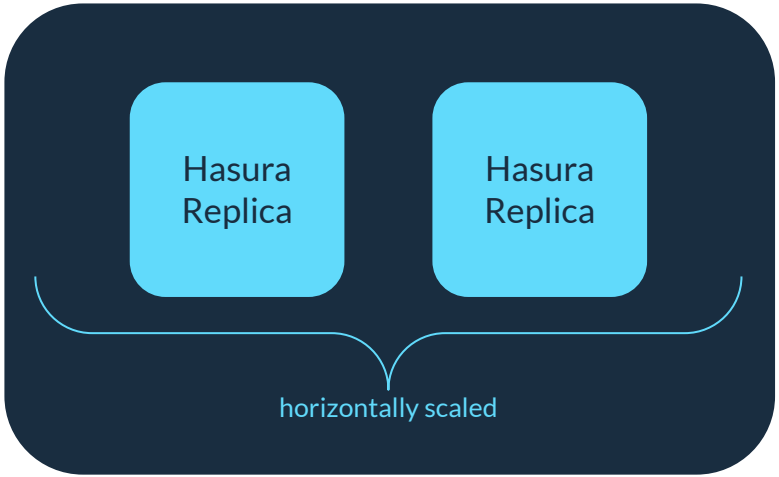
1. R&D + Implementation
2. Deployment
3. Maintenance

How was it maintaining Hasura in Prod?

Boring.

And that’s a good thing.

Database Maintenance = autopilot

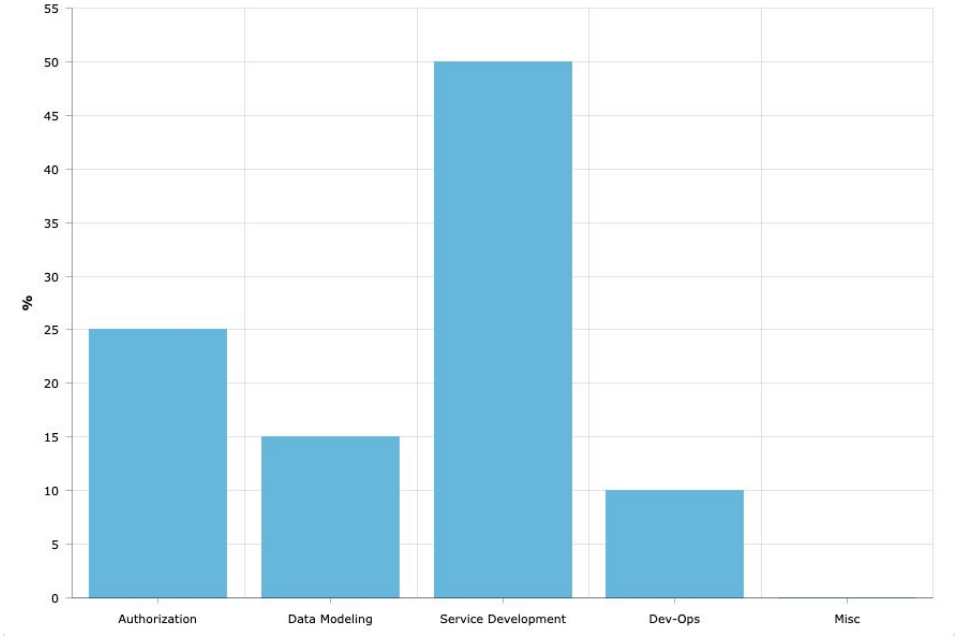


- ✔ Hasura horizontally scaled in K8
- ✔ EC2 auto-scaled scaled K8 Nodes
- ✔ RDS Multi-AZ Deployment
- ✔ Aurora Compute Scaling

How did Hasura affect eng day-to-day?

Backend Eng Team's Time

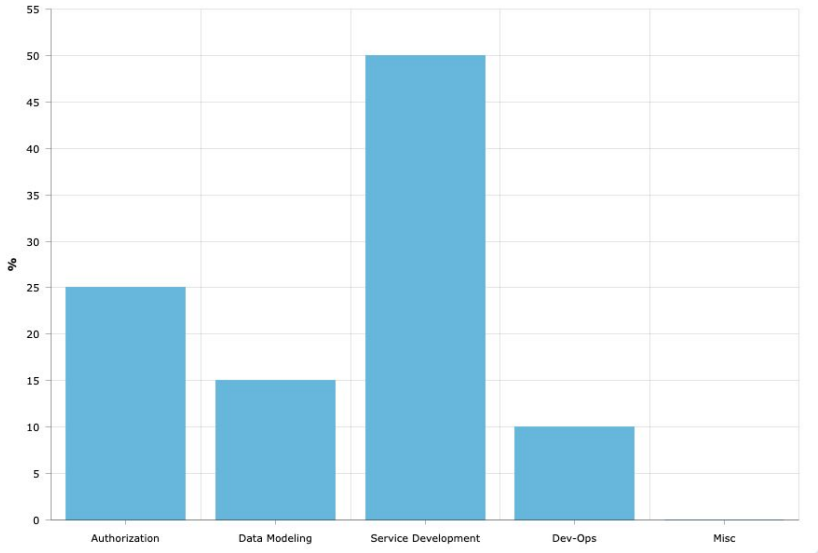
Before Hasura



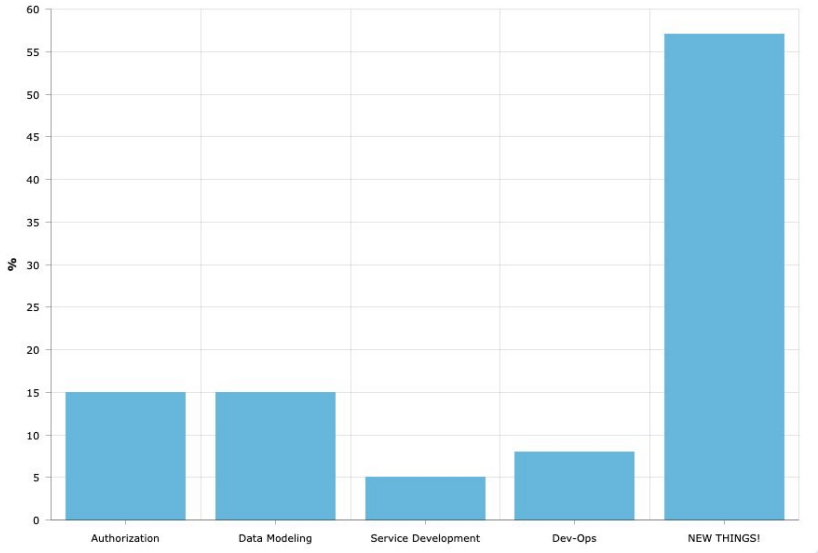
% determined by analysis of Git commit history in respective projects

Backend Eng Team's Time

Before Hasura



After Hasura



57% time freed to work on new things

- Continue migration to Golang
- Spend more time on R&D
- Build product faster. Less backend bottlenecks for Frontend Eng

Adopting Hasura in Retrospect

- Frontend:
  - No more backend bottlenecks
  - Heightened (yet secure) access to database
- Backend:
  - More time to focus on new features, tech debt, etc
- Ops:
  - Less services to maintain

WorkClout

- Enterprise Users
- Multi-tenant
- Real-time
- SQL-based & Scalable

iOS

WEB

3 Months to G.T.M.

QUESTIONS?

RICHARD GIRGES -
RICHARDGIRGES@GMAIL.COM

@richardgirges

Find this deck here:
GITHUB.COM/RICHARDGIRGES/TECHTALKS