Graphql

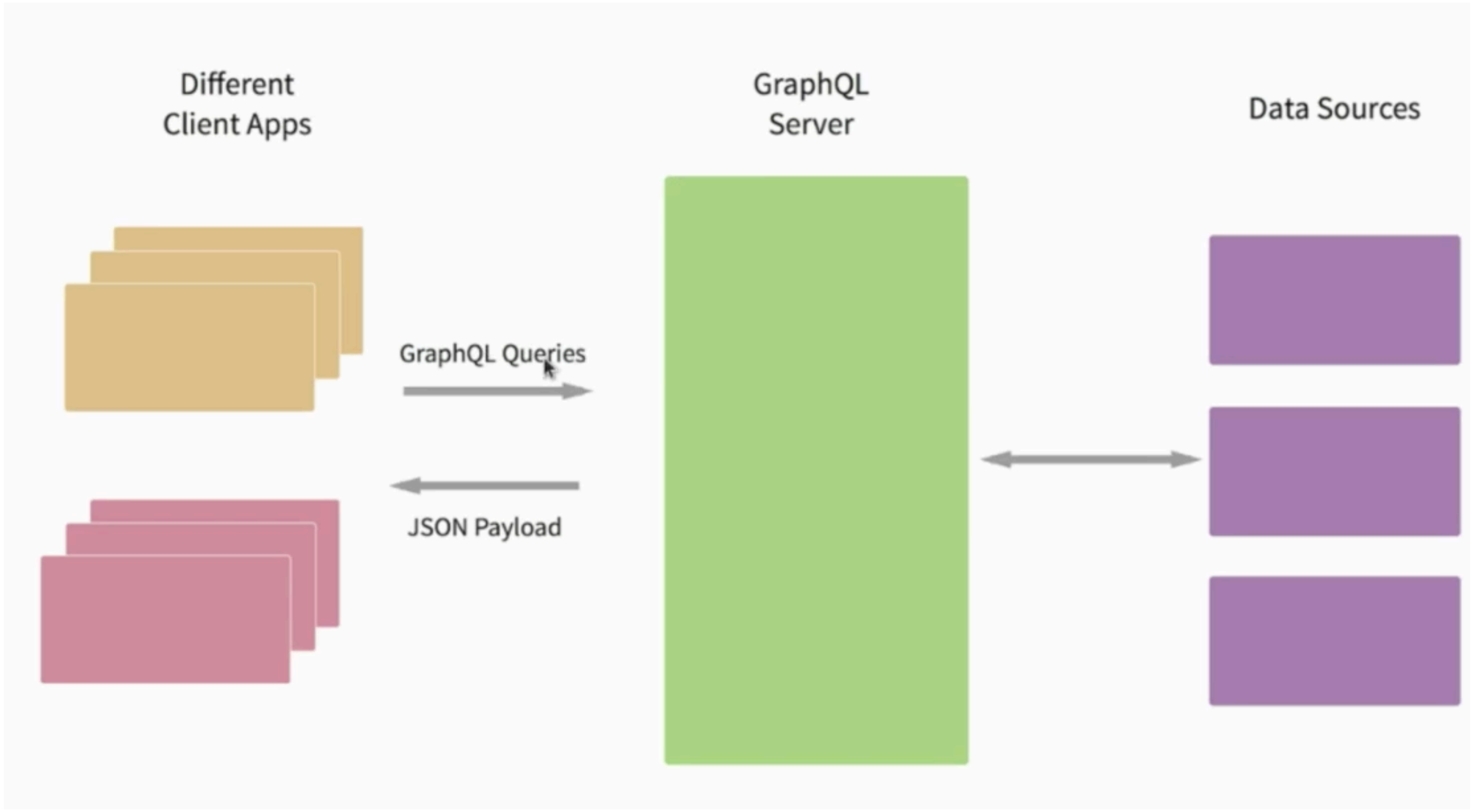
REST

- Pequenos problemas
  - Vários endpoints
  - Url com vários parâmetros
  - Dados de respostas (não configurável)
  - Impacto em tráfico e latência

Graphql

- Facebook 2012
- 2015 liberado como Open Source
- Uma liguagem de consulta
  - Cliente pede o que precisa e servidor retorna
  - Sem conceito de recursos (endpoint)

Graphql


Different Client Apps
GraphQL Server
Data Sources
GraphQL Queries
JSON Payload

G

```
query {
  user(name:"Joao") {
    name
    age
    city
    knowledge {
      language
    }
  }
}
```

```
{
  "data": {
    "user": {
      "name": "Joao",
      "age": 22,
      "city": "Campina Grande",
      "knowledge": [
        {
          "language": "Javascript"
        },
        {
          "language": "Java"
        },
        {
          "language": "Python"
        }
      ]
    }
  }
}
```

Variáveis

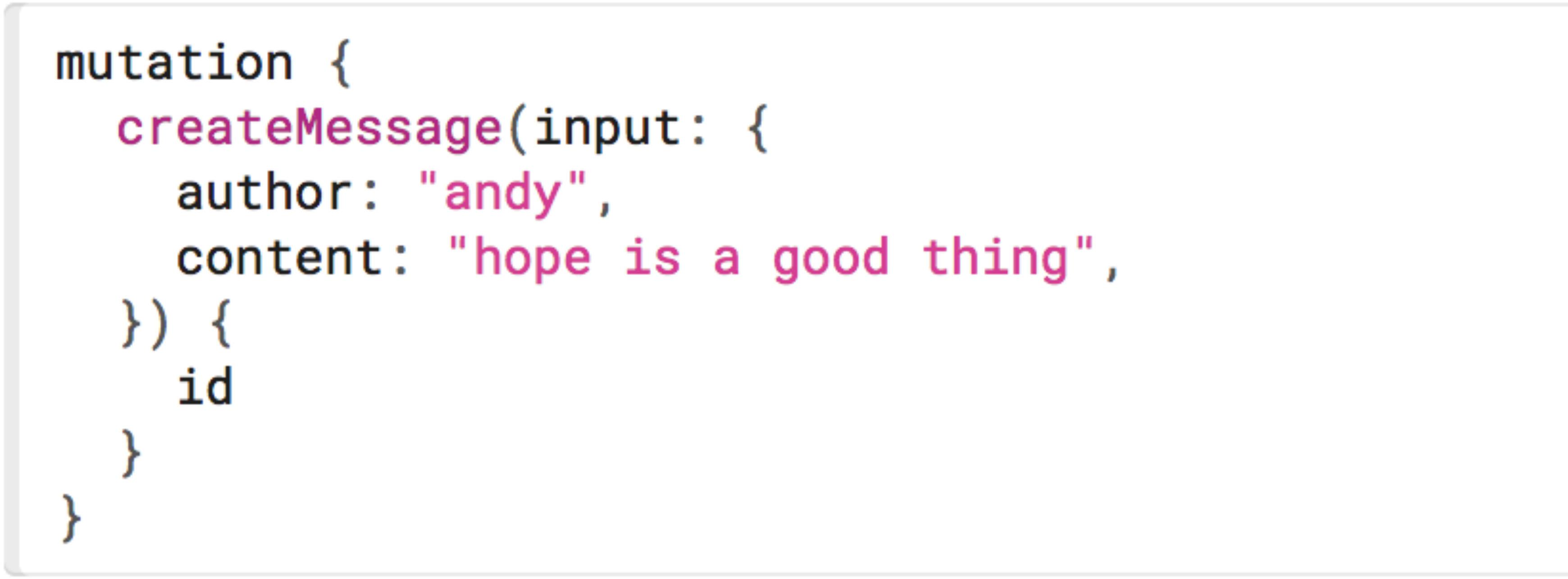
```
query($numberOfBuilds:Int!) {
  viewer {
    user {
      name
      builds(first: $numberOfBuilds) {
        edges {
          node {
            number
            branch
            message
          }
        }
      }
    }
  }
}
```

Variáveis

```
curl 'https://graphql.buildkite.com/v1' \
  -H 'Authorization: Bearer xxxxxxx' \
  -d '{
    "query": "query($numberOfBuilds:Int) { viewer { user { name
builds(first: $numberOfBuilds) { edges { node { number branch
message } } } } } }",
    "variables": { "numberOfBuilds": 1 }
  }'
```

Mutations

- POST, PUT, PATCH and DELETE



```
mutation {
  createMessage(input: {
    author: "andy",
    content: "hope is a good thing",
  }) {
    id
  }
}
```

Schema

```
// Construct a schema, using GraphQL schema language
var schema = buildSchema(`
  input MessageInput {
    content: String
    author: String
  }

  type Message {
    id: ID!
    content: String
    author: String
  }

  type Query {
    getMessage(id: ID!): Message
  }

  type Mutation {
    createMessage(input: MessageInput): Message
    updateMessage(id: ID!, input: MessageInput): Message
  }
`);
```

NodeJS

```
var root = {
  getMessage: function ({id}) {
    if (!fakeDatabase[id]) {
      throw new Error('no message exists with id ' + id);
    }
    return new Message(id, fakeDatabase[id]);
  },
  createMessage: function ({input}) {
    // Create a random id for our "database".
    var id = require('crypto').randomBytes(10).toString('hex');

    fakeDatabase[id] = input;
    return new Message(id, input);
  },
  updateMessage: function ({id, input}) {
    if (!fakeDatabase[id]) {
      throw new Error('no message exists with id ' + id);
    }
    // This replaces all old data, but some apps might want partial update.
    fakeDatabase[id] = input;
    return new Message(id, input);
  },
};
```

NodeJS

```
var app = express();
app.use('/graphql', graphqlHTTP({
  schema: schema,
  rootValue: root,
  graphiql: true,
}));
app.listen(4000, () => {
  console.log('Running a GraphQL API server at localhost:4000/graphql');
});
```

NodeJS

```
var author = 'andy';
var content = 'hope is a good thing';
var xhr = new XMLHttpRequest();
xhr.responseType = 'json';
xhr.open("POST", "/graphql");
xhr.setRequestHeader("Content-Type", "application/json");
xhr.setRequestHeader("Accept", "application/json");
xhr.onload = function () {
  console.log('data returned:', xhr.response);
}
var query = `mutation CreateMessage($input: MessageInput) {
  createMessage(input: $input) {
    id
  }
}`;
xhr.send(JSON.stringify({
  query: query,
  variables: {
    input: {
      author: author,
      content: content,
    }
  }
}));
```