



# Serverless is more than your favourite cloud!

How to do Serverless well!



# SUMMARY OF TOPICS

## MAIN POINTS COVERED

Repo and Environment Variables

CI/CD Pipeline

Serverless Framework

Appsync – Apollo GraphQL

VTL and DynamoDB

GraphQL to test

Nextjs – Zeit

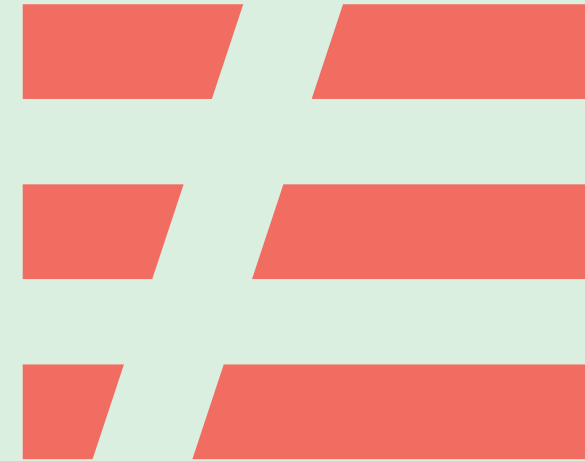
React Semantic UI

Microservices with Appsync

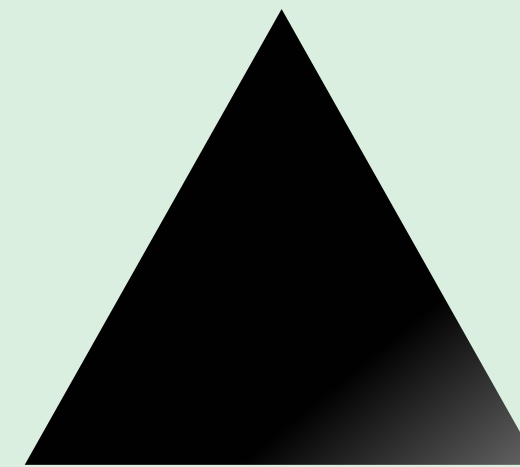
Auth0 with Appsync and Nextjs

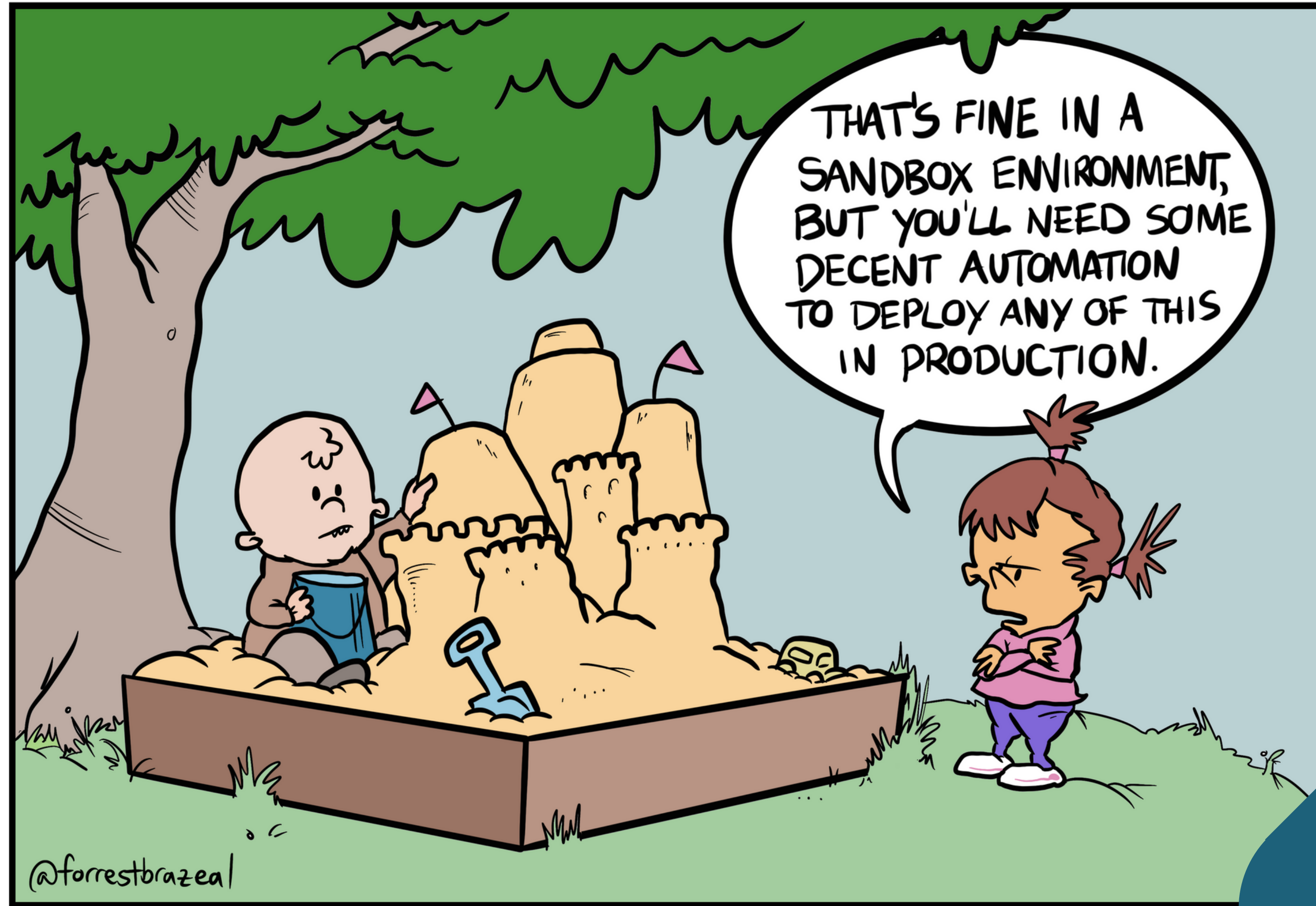
Using Apollo React-Hooks

# ARCHITECTURE



~~NEXT~~.JS







# Don't store credentials in your repo



SAYS EVERYONE

# GETTING SETUP



Github



Deployment Bucket



SSM Parameter Store





```
Resources:
```

```
  auth0Issuer:
```

```
    Type: "AWS::SSM::Parameter"
```

```
    Properties:
```

```
      Name: !Sub "${ApplicationName}/${Environment}/auth0/issuer"
```

```
      Type: "String"
```

```
      Value: "hello"
```

```
      Description: "Auth0 issuer - the domain."
```

```
      Tags:
```

```
        "Environment": !Ref Environment
```

```
        "app": !Ref ApplicationName
```



# TRIGGERING DEPLOYMENTS



Code Change /  
New Commit



Triggers  
CodePipeline



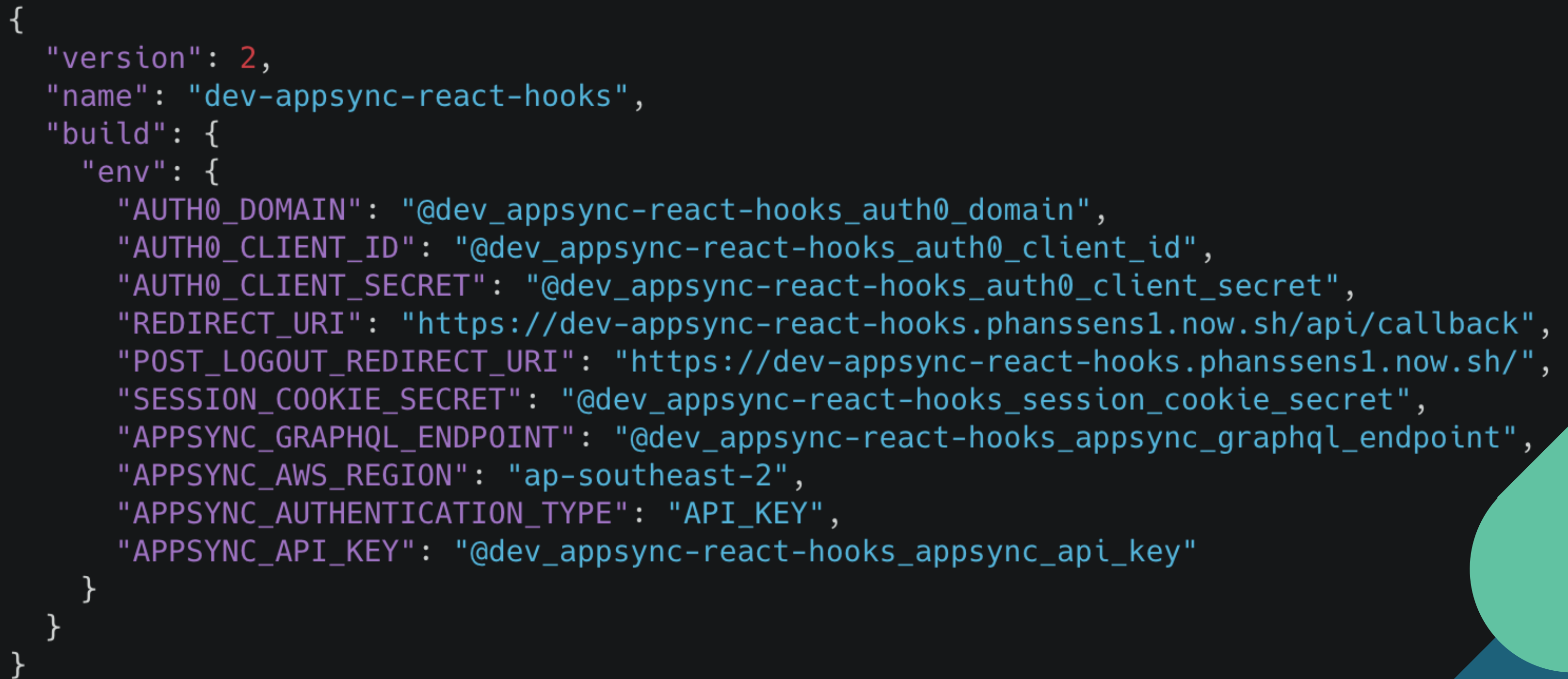
CodeBuild deploys  
SLS and Zeit



```
build:
```

```
  commands:
```

- echo Entered the build phase...
- echo Build started on `date`
- pwd
- # Deploy backend services
- cd backend/services
- npm i --save
- sls deploy --stage dev
- # Deploy frontend to Zeit
- cd ../../frontend
- npm i --save
- yarn build
- now -A config/dev/now.json -t \$now\_token




```
{
  "version": 2,
  "name": "dev-appsync-react-hooks",
  "build": {
    "env": {
      "AUTH0_DOMAIN": "@dev_appsync-react-hooks_auth0_domain",
      "AUTH0_CLIENT_ID": "@dev_appsync-react-hooks_auth0_client_id",
      "AUTH0_CLIENT_SECRET": "@dev_appsync-react-hooks_auth0_client_secret",
      "REDIRECT_URI": "https://dev-appsync-react-hooks.phanssens1.now.sh/api/callback",
      "POST_LOGOUT_REDIRECT_URI": "https://dev-appsync-react-hooks.phanssens1.now.sh/",
      "SESSION_COOKIE_SECRET": "@dev_appsync-react-hooks_session_cookie_secret",
      "APPSYNC_GRAPHQL_ENDPOINT": "@dev_appsync-react-hooks_appsync_graphql_endpoint",
      "APPSYNC_AWS_REGION": "ap-southeast-2",
      "APPSYNC_AUTHENTICATION_TYPE": "API_KEY",
      "APPSYNC_API_KEY": "@dev_appsync-react-hooks_appsync_api_key"
    }
  }
}
```

# DEPLOYING A MICROSERVICE






```
appSync:
- name: ${self:custom.service.private-appsync}
  schema: src/AppSync/privateSchema.graphql
  authenticationType: OPENID_CONNECT
  openIdConnectConfig:
    issuer: ${ssm:/appsync-react-hooks/dev/auth0/issuer}
    clientId: ${ssm:/appsync-react-hooks/dev/auth0/clientId}
    iatTTL: 86400
    authTTL: 7200
  serviceRole: AppSyncRole
  logConfig:
    loggingRoleArn: { Fn::GetAtt: [AppSyncRole, Arn] } # Where AppSyncLoggingServiceRole is a role with
CloudWatch Logs write access
    level: ERROR # Logging Level: NONE | ERROR | ALL
  dataSources:
    - type: AMAZON_DYNAMODB
      name: diaryAppSync
      description: 'diary App Sync'
      config:
        tableName: { Ref: DiaryTable } # Where MyTable is a dynamodb table defined in Resources
        serviceRoleArn: { Fn::GetAtt: [AppSyncRole, Arn] } # Where AppSyncDynamoDBServiceRole is an IAM role
defined in Resources
  mappingTemplates:
    - dataSource: diaryAppSync
      type: Mutation
      field: createEvent
      request: "diary/createEvent.txt"
      response: "JSONResponse.txt"
```



```
type Mutation {  
  
    # Delete diaryEntry  
    deleteEvent(  
        user_id: ID!  
        date: String!  
    ): Event  
  
    # Events  
    createEvent(  
        user_id: String!  
        date: String!  
        category: String!  
        event: String!  
    ): PutEvent  
}  
  
type Query {  
    getEvents(user_id: String!, date: String!): GetEvents  
}
```



```
# Events  
  
type PutEvent {  
    user_id: String!  
    date: String!  
    category: String!  
    event: String!  
}  
  
type GetEvents {  
    user_id: String!  
    date: String!  
    diary_entry: [Event]  
}  
  
type Event {  
    created_at: String  
    created_epoch: Int  
    category: String!  
    event: String!  
}  
  
schema {  
    query: Query  
    mutation: Mutation  
}
```

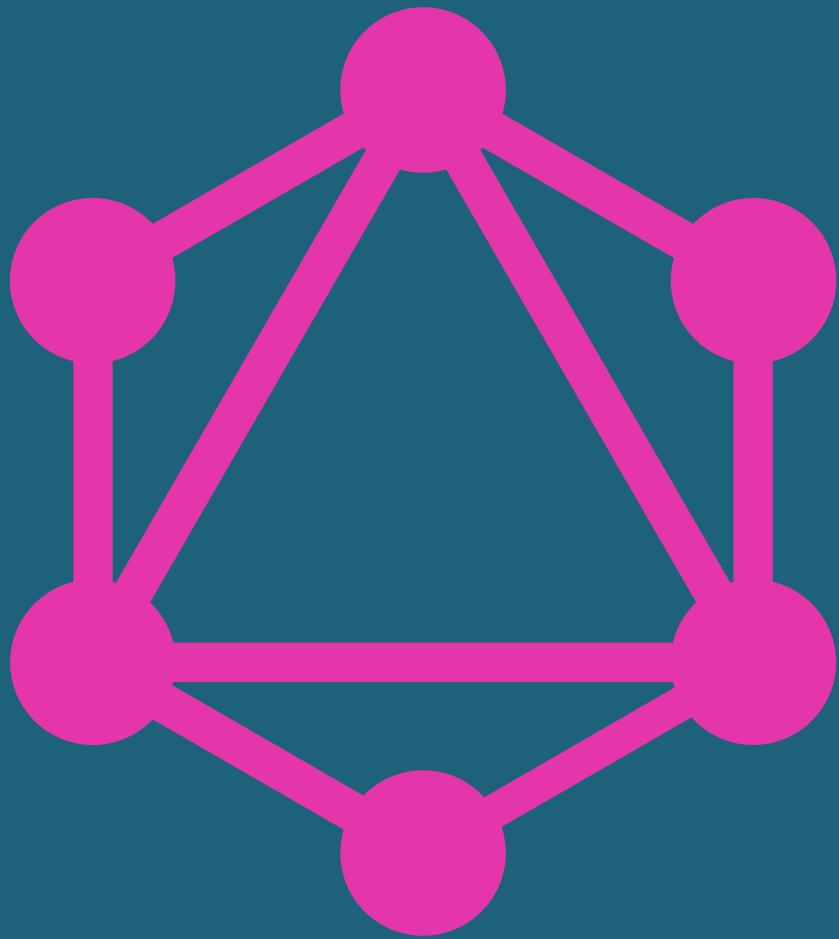




```
{
  "version" : "2017-02-28",
  "operation" : "UpdateItem",
  "key" : {
    "user_id" : { "S" : "${context.arguments.user_id}" },
    "date" : { "S" : "${context.arguments.date}" }
  },
  "update" : {
    "expression" : "SET diary_entry = list_append(if_not_exists(diary_entry, :emptyList), :newDiaryEntry)",
    "expressionValues" : {
      ":emptyList" : { "L" : [] },
      ":newDiaryEntry" : { "L" : [
        { "M" : {
          "created_at" : { "S" : "$util.time.nowISO8601()" },
          "created_epoch" : { "N" : "$util.time.nowEpochMilliseconds()" },
          "category" : { "S" : "${context.arguments.category}" },
          "event" : { "S" : "${context.arguments.event}" }
        }
      ] }
    }
  }
}
```



# TESTING THE ENDPOINT



GetDiaryEntries

GraphQL Endpoint

Method POST Edit HTTP Headers

GraphiQL ▶ Prettify History

```
1 query ($user_id: String!, $date: String!) {
2   getEvents (user_id: $user_id, date: $date) {
3     user_id
4     date
5     diary_entry {
6       category
7       event
8     }
9   }
10 }
```

QUERY VARIABLES

```
1 {
2   "user_id": "auth0|sample-user",
3   "date": "2020-03-01"
4 }
```

```
{
  "data": {
    "getEvents": {
      "user_id": "auth0|sample-user",
      "date": "2020-03-01",
      "diary_entry": [
        {
          "category": "coding",
          "event": "wrote some nodejs"
        },
        {
          "category": "beverages",
          "event": "drank coffee"
        },
        {
          "category": "coding",
          "event": "wrote some more code"
        }
      ]
    }
  }
}
```

< Schema Mutation ×

No Description

FIELDS

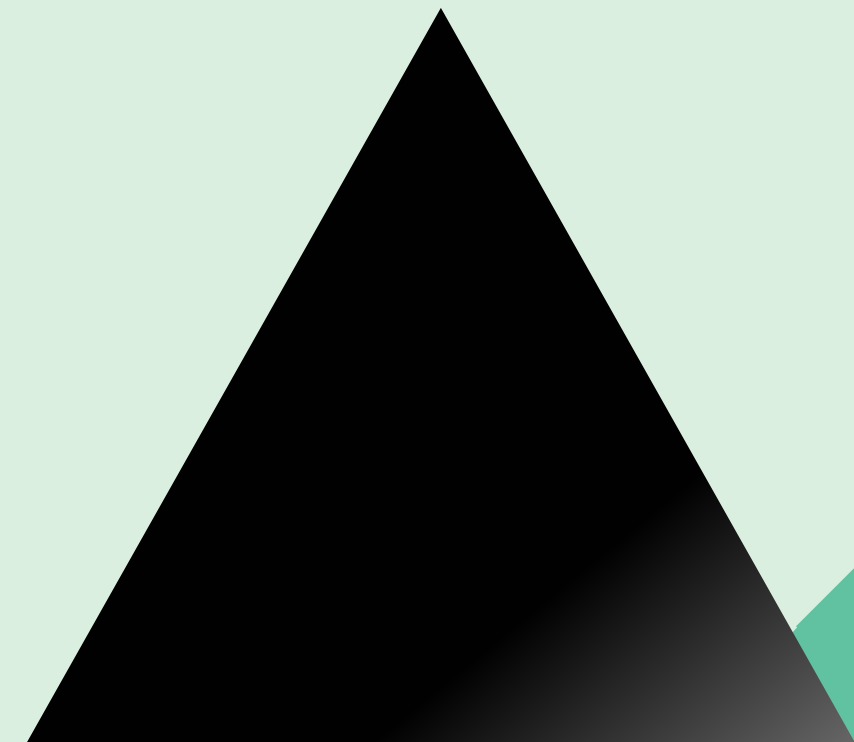
```
deleteEvent(user_id: ID!, date: String!): Event
Delete diaryEntry

createEvent(
  user_id: String!
  date: String!
  category: String!
  event: String!
): PutEvent

Weather
```

# THE FRONTEND

NEXT.js



# WHAT IS NEXTJS

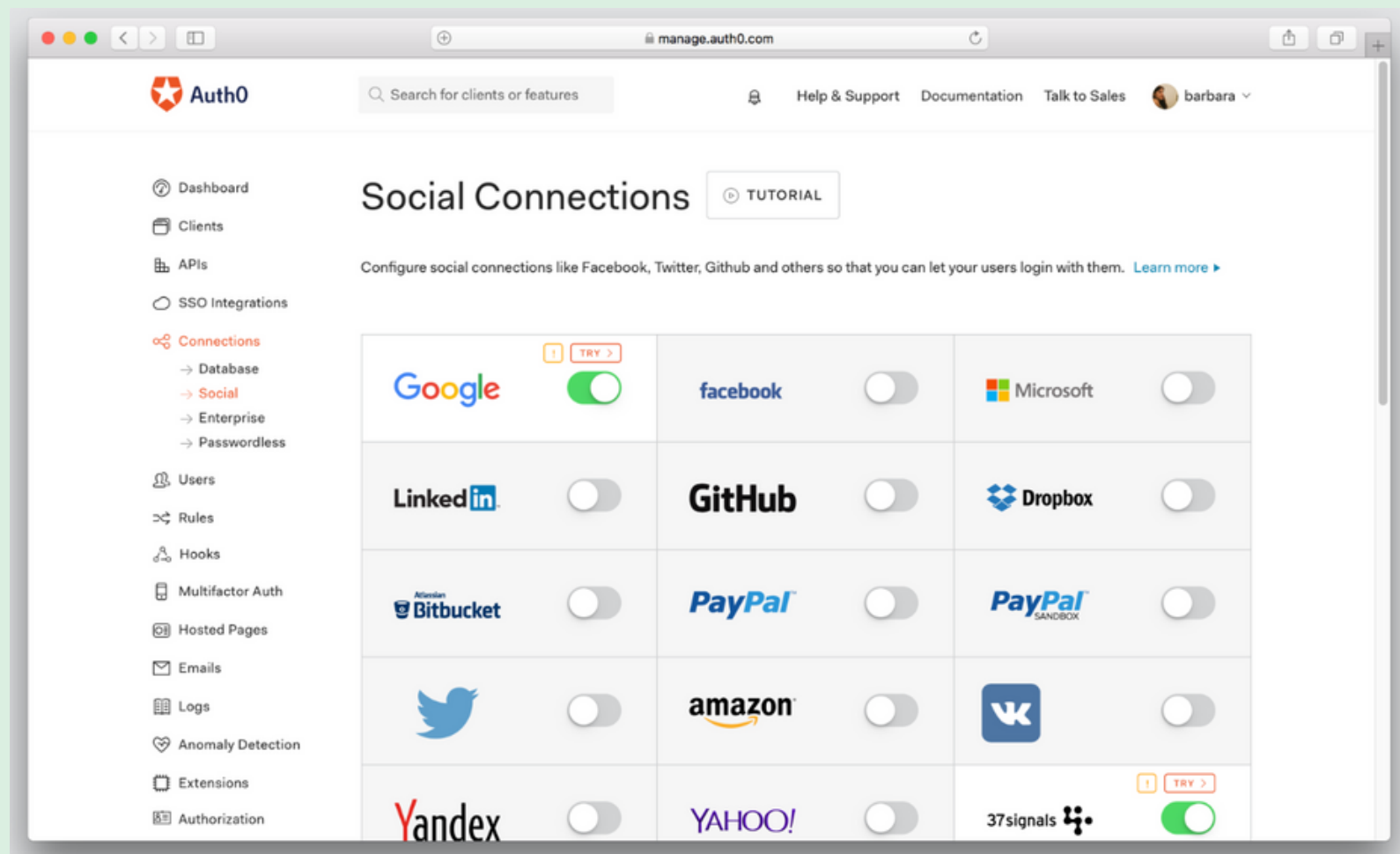


- Server side rendering
- Zero setup
- Can also export static content
- CSS in JS

# WHAT IS AUTH0



- Handles logins for you
- Allows you to use social providers with ease
- Has a world class admin panel



# WHAT IS ZEIT NOW

- Cloud Platform to deploy Server Side Rendered Web Apps
- Integration with github, gitlab, bitbucket
- Can deploy Vuejs, Gatsby, Svelte apps



# ADDING PACKAGES



# REACT SEMANTIC UI



```
import { Link } from 'react-router-dom'  
import { Button } from 'semantic-ui-react'
```

jsx

```
// 💡 `to` prop is not handled in `Button` and will be passed to `Link` component
```

```
<Button as={Link} to="/home">
```

```
  To homepage
```

```
</Button>
```





The background is a deep blue gradient with a large, glowing purple and blue planet in the center. Several stylized fish are scattered around: a pufferfish in the top left, a striped fish in the top right, a small fish with bubbles in the middle right, a stingray in the bottom left, a fish with a lightbulb in the bottom center, and a simple fish in the bottom right. A vertical line with a pink circle is positioned near the center.

# APOLLO

CLIENT

HOOKS

# APOLLO REACT-HOOKS

```
const [deleteEvent] = useMutation(Delete, {  
  refetchQueries: ["getEvents"],  
});  
const [createEvent] = useMutation(mutation, {  
  refetchQueries: ["getEvents"],  
});
```

# Repo

<https://github.com/phanssens1/appsync-react-hooks>



# CONTACT ME

TWITTER

@petehanssens

GITHUB

<https://github.com/phanssens1>

EMAIL

[sydney@serverlessdays.io](mailto:sydney@serverlessdays.io)