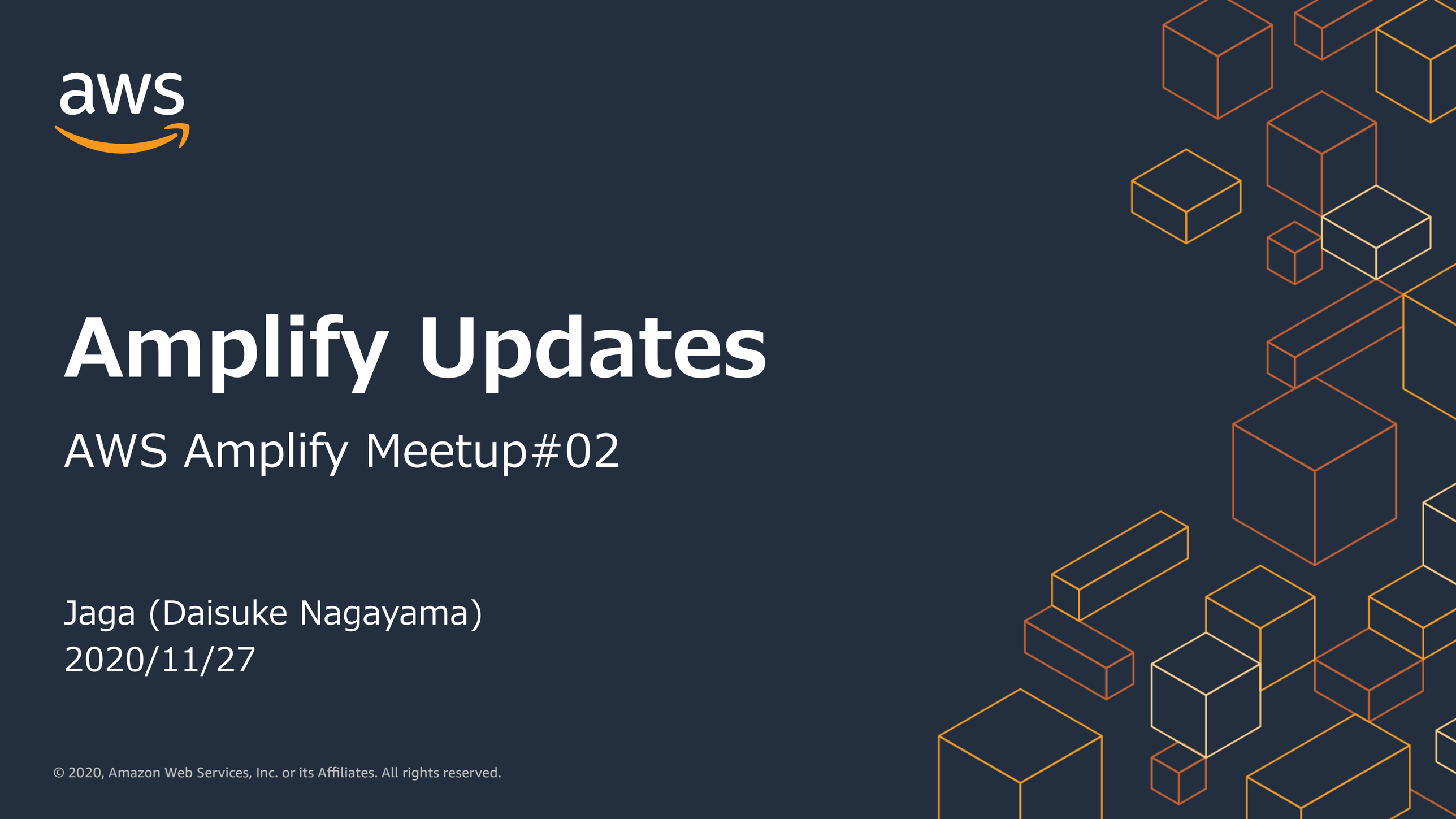


Amplify Updates

AWS Amplify Meetup#02

Jaga (Daisuke Nagayama)
2020/11/27

Jaga (Daisuke Nagayama) @jagaimogmog

Startup Solutions Architect
Amazon Web Service Japan K.K.

> Health Care Startupでエンジニア・事業開発
> AWSでStartupの技術支援・Mobile

AWS Amplify

Amazon Chime SDK

Amazon Pinpoint

Amplify Meetup #01 以降*の主要アップデート

11/22	Amplify CLI で S3 バケットと DynamoDB テーブルをインポート可能に
11/4	Amplify Console が Performance Mode をサポート
10/17	Amplify CLI で Cognito User Pools & Identity Pools を インポート可能に
9/15	Amplify JavaScript が SSR をサポート
8/31	Amplify CLI がプロジェクト単位でのリソースのタグづけをサポート
9/1	Amplify Android が RxJava をサポート
8/20	Amplify Flutter の Developer Preview が利用可能に
8/17	Amplify iOS が Swift Combine をサポート

* 2020/7/31 以降

aws

Amplify CLI Updates

Amplify CLI で S3 バケットと DynamoDB テーブルをインポート可能に

Amplify CLI v4.30.0 (10/17)

```
$ amplify import storage
? Please select from one of the below mentioned services: DynamoDB table - NoSQL Database
✔ Select the DynamoDB Table you want to import: · Comment-bwmpu2kmsbfi3biwt6kywnvnnm-dev

✅ DynamoDB Table 'Comment-bwmpu2kmsbfi3biwt6kywnvnnm-dev' was successfully imported.

Next steps:
- This resource can now be accessed from REST APIs (‘amplify add api’) and Functions (‘amplify add function’)
```

ユースケース

- 既存アプリの S3 バケット・DynamoDB テーブルを Amplify でつくるアプリでも使いたい
- Amplify を使う一方で、S3 バケットや DynamoDB テーブルは Amplify 外のライフサイクルで管理したい

注意点

- Cognito 認証ユーザーに S3/DynamoDB アクセス用の IAM の権限設定が必要
- インポート した DynamoDB テーブルは APIカテゴリの REST API や function カテゴリの Lambda 関数からはアクセス可能な一方、API カテゴリの GraphQL API で利用する DynamoDB としては利用不可

https://aws.amazon.com/blogs/mobile/use-an-existing-s3-bucket-for-your-amplify-project/

aws

Cognito User Pools & Identity Pools を Amplify CLI でインポート可能に

Amplify CLI
v4.33.0 (11/22)

```
$ amplify import auth
Initializing new Amplify CLI version...
Done initializing new version.
Scanning for plugins...
Plugin scan successful
Using service: Cognito, provided by: awscloudformation
✔ What type of auth resource do you want to import? · Cognito User Pool only
✔ Select the User Pool you want to import: · us-east-1_HawXs09vB
✔ Only one Web app client found: 'postapc6ee4a23_app_clientWeb' was automatically selected.
✔ Only one Native app client found: 'postapc6ee4a23_app_client' was automatically selected.
✔ Federated identity providers are not configured, no OAuth configuration needed.

✅ Cognito User Pool 'postappc6ee4a23_userpool_c6ee4a23-prod' was successfully imported.
```

ユースケース

- 既存アプリのユーザー基盤をAmplifyでつくるアプリでも使いたい
- 既存アプリを徐々に Amplify に移行していきたい
- Amplify を使う一方で、重要なユーザー基盤は Amplify の外のライフサイクルで管理したい

https://aws.amazon.com/blogs/mobile/use-existing-cognito-resources-for-your-amplify-api-storage-and-more/

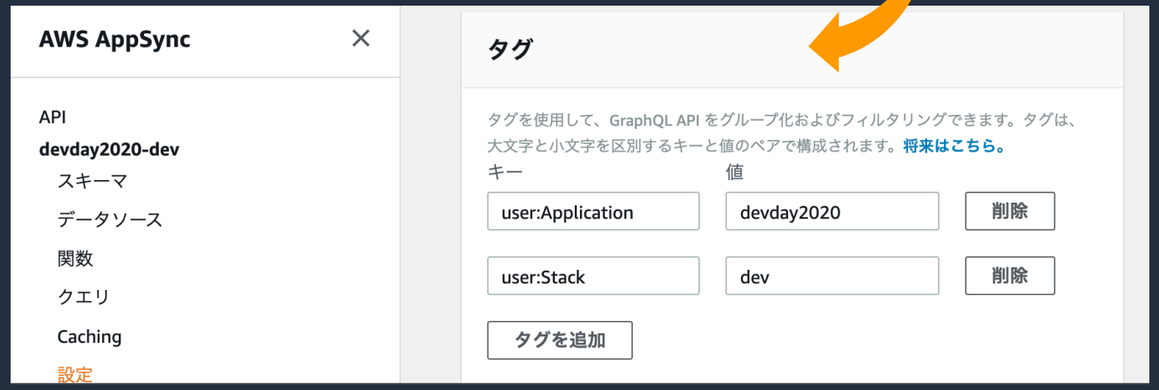
aws

Amplify CLI がプロジェクト単位でのリソースのタグづけをサポート

Amplify CLI v4.28.0 (8/31)

```
[
  {
    "Key": "user:Stack",
    "Value": "{project-env}"
  },
  {
    "Key": "user:Application",
    "Value": "{project-name}"
  }
]
```

$ amplify push



概要

- Amplify CLI v4.28.0 以降で、プロジェクト単位でのリソースへのタグ付け機能が利用可能に
- 請求時の計算の簡略化

使い方

- tags.json の配列に追加することで最大50までのタグを使用可能
- {project-env} と {project-name} 変数は $ amplify push 時に代入される

注意事項

- 4.28.0 以前のバージョンで作成されたプロジェクトの場合、tags.json ファイルを手動で作成する必要あり

https://docs.amplify.aws/cli/usage/tags

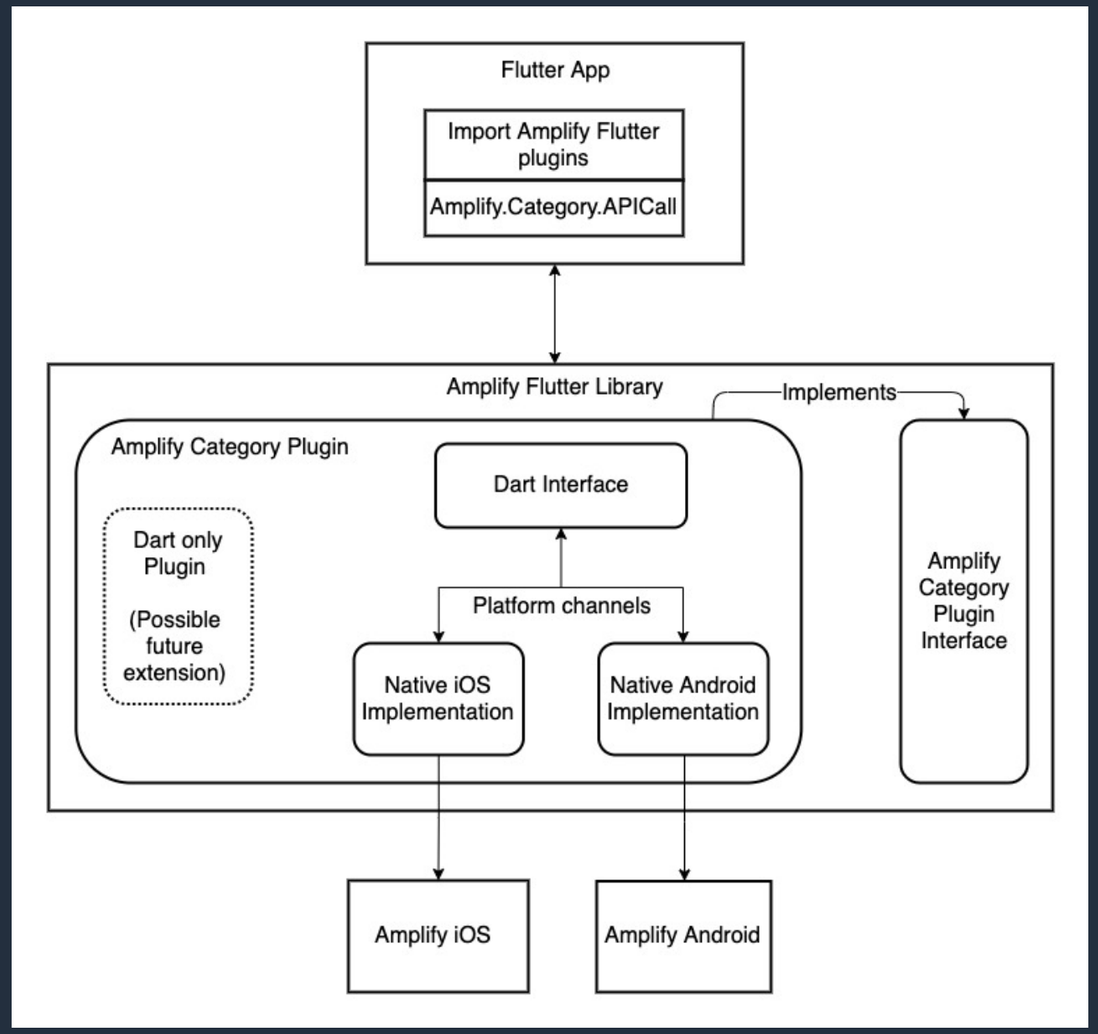
Amplify Flutter Developer Preview

Amplifiy Flutter Developer Preview (1)

- 現状 Auth, Storage, Analytics の3カテゴリが使用可能
- 将来的には API (REST/GraphQL), DataStore, Predictions, Escape Hacthe が実装予定
- **Preview のため本番環境での利用は非推奨**

https://aws.amazon.com/blogs/mobile/announcing-aws-amplify-flutter-developer-preview/

Amplifiy Flutter Developer Preview (2)



- Dart インタフェースから Platform Channels を通じて Amplify iOS, Android を呼び出す
- 将来的には Amplify iOS/Android でなく、Dartで完結するライブラリ機能も実装可能なデザイン

https://aws.amazon.com/blogs/mobile/announcing-aws-amplify-flutter-developer-preview/

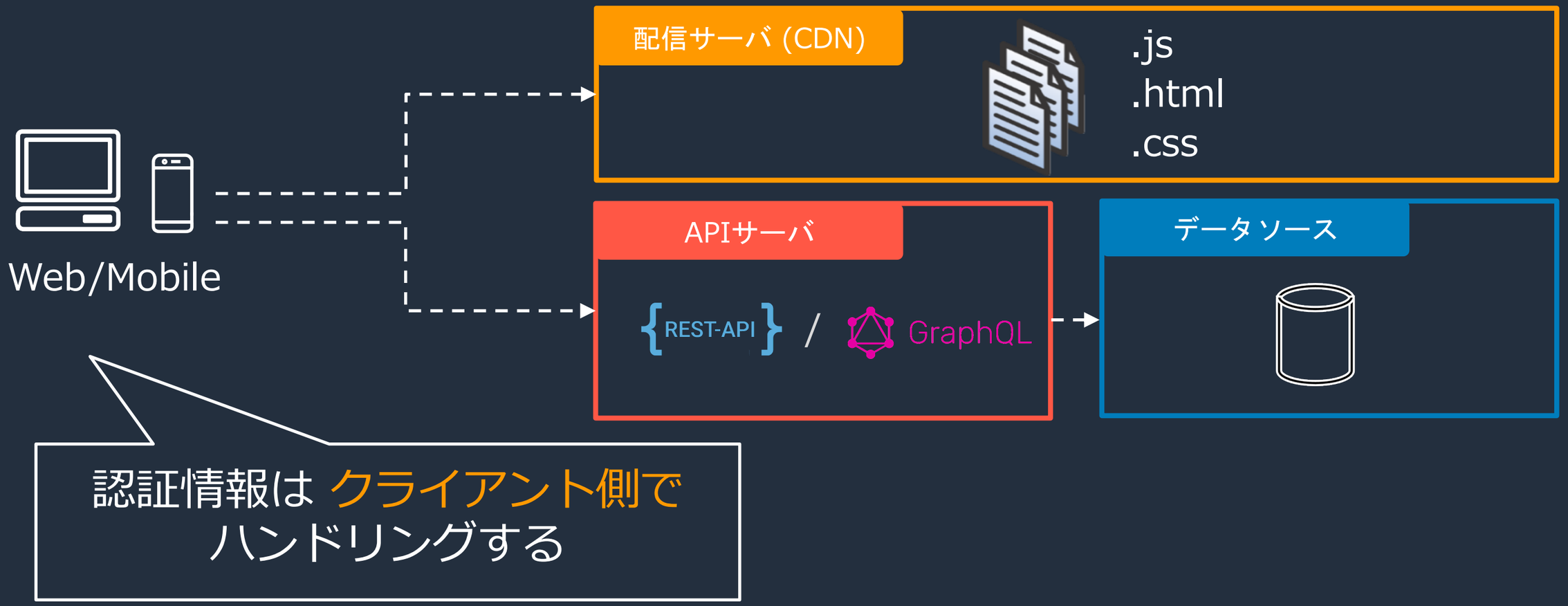
Amplifiy Flutter Developer Preview (3)

- Tutorial: https://docs.amplify.aws/start/q/integration/flutter
- Workshop: https://aws.amazon.com/jp/getting-started/hands-on/build-flutter-app-amplify/
- Document: https://docs.amplify.aws/lib/q/platform/flutter
- GitHub Repository: https://github.com/aws-amplify/amplify-flutter

aws

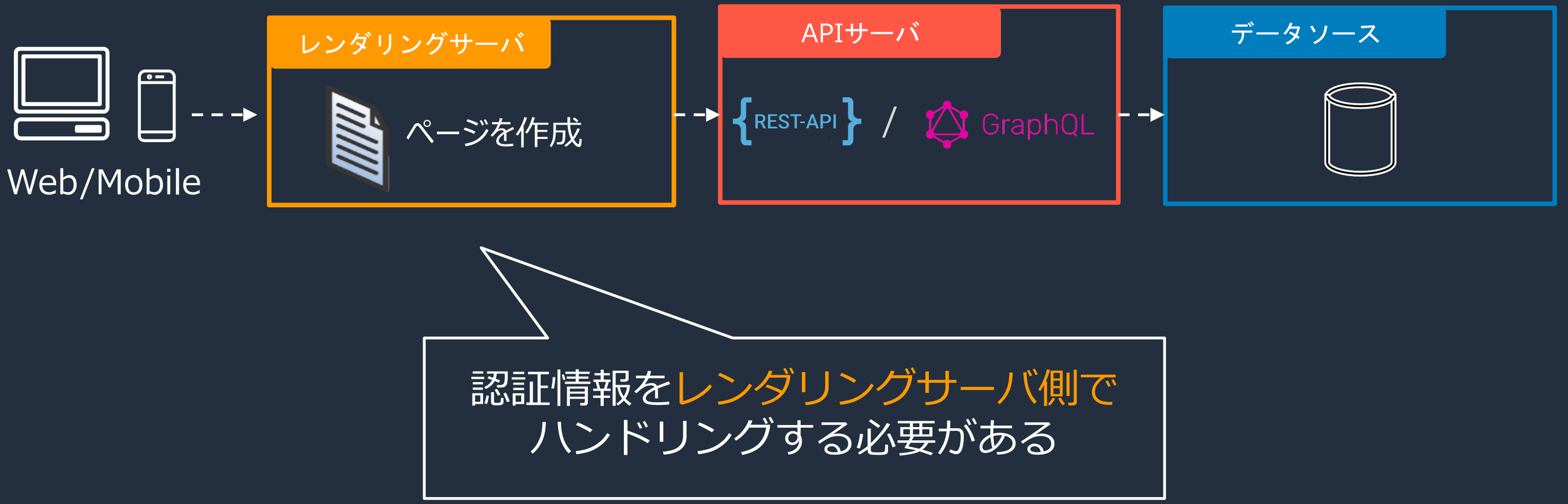
Server Side Rendering (SSR) Support

Single Page Application (SPA) の構成



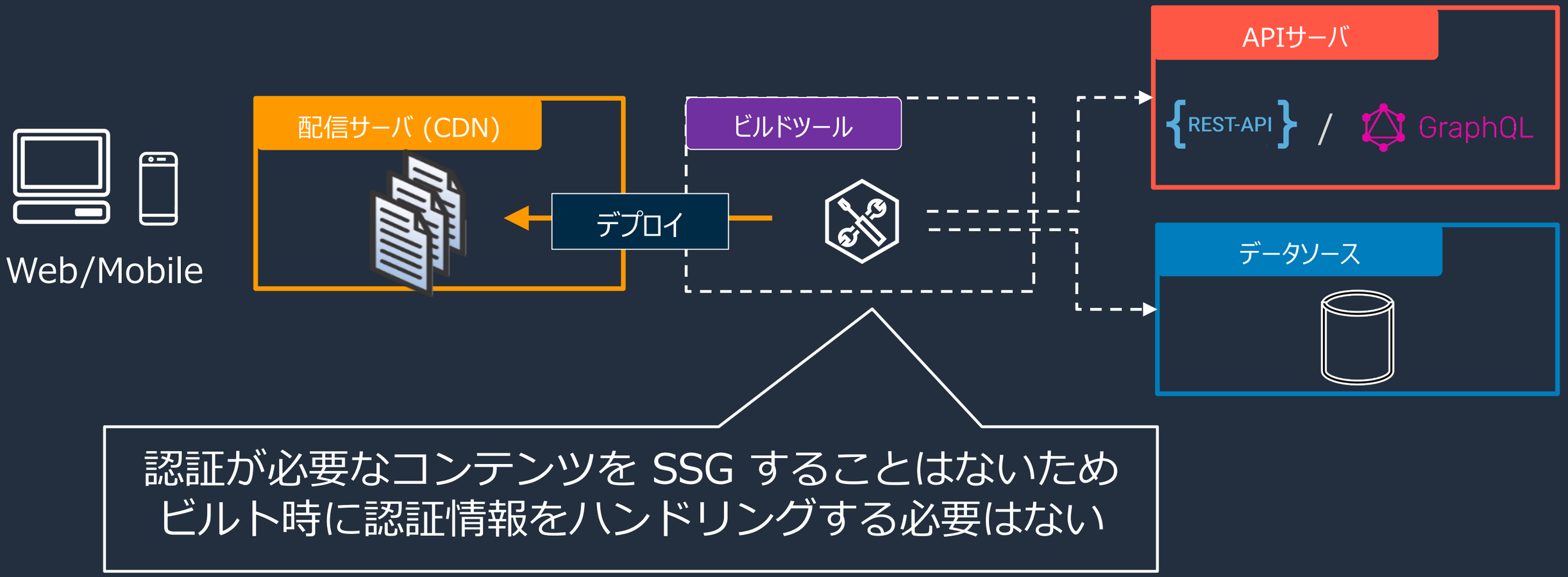
フロントエントエンジニアのためのAWSアーキテクチャ より一部引用

Server Side Rendering (SSR)の構成



フロントエントエンジニアのためのAWSアーキテクチャ より一部引用

Static Site Generator (SSG) を用いた構成



フロントエントエンジニアのためのAWSアーキテクチャ より一部引用

Amplify JavaScript が SSR をサポート (1)

Amplify JS
v3.1.0 (9/3)

NUXTJS

- Amplify JavaScript は認証情報のトークンをクライアントサイドでハンドリングするため、サーバーサイドでの利用が難しかった
- Amplify JavaScript v3.1.0 以降、Next.js/Nuxt.js で SSR 利用時でも Auth や API が動くように
- SEO や OGP, 速報性が必要なアプリケーションなど SSR が必要なケースでも Amplify が利用可能に

https://aws.amazon.com/blogs/mobile/ssr-support-for-aws-amplify-javascript-libraries/

aws

Amplify JavaScript が SSR をサポート (2)

Amplify JS
v3.1.0 (9/3)

```javascript
import { API } from 'aws-amplify';
import { listMovies } from '../src/graphql/queries';

export default function Movies(props) {
  const { movies } = props;
  return (
    <div>
      {
        movies.map(movie => (
          <div key={movie.id}>
            <h3>{movie.name}</h3>
            <span>{movie.description}</span>
          </div>
        ) )
      }
    </div>
  )
}

export async function getStaticProps() {
  const movieData = await API.graphql({ query: listMovies });
  return {
    props: {
      movies: movieData.data.listMovies.items
    }
  }
}
```

- Next.js では Render 用関数(左図 Movies)は props 引数を受け取りレンダリングを行う
- あらかじめビルドしておく (SSG) 場合は、**getStaticProps** 関数を呼び出すことで、props の中身を定義できる

https://aws.amazon.com/blogs/mobile/ssr-support-for-aws-amplify-javascript-libraries/

Amplify JavaScript が SSR をサポート (2)

Amplify JS
v3.1.0 (9/3)

```
// pages/_app.js
import { Amplify } from "aws-amplify";
import awsExports from "../src/aws-exports";
Amplify.configure({ ...awsExports, ssr: true });
```

```
export async function getServerSideProps(context) {
  const { API } = withSSRContext(context)
  let movieData
  try {
    movieData = await API.graphql({
      query: listMovies,
      authMode: "AMAZON_COGNITO_USER_POOLS"
    });
    console.log('movieData: ', movieData)
  } catch (err) {
    console.log("error fetching movies: ", err)
  }
  return {
    props: {
      movies: movieData ? movieData.data.listMovies.items : null
    }
  }
}
```

- Amplify.configureでssr:trueを指定
- ユーザーアクセス時に動的に SSR する場合、Render 用関数に渡す引数を **getServerSideProps** 関数で定義することで動的に props の生成とレンダリングを行う
- getServerSideProps 関数内で、Amplify の提供する **withSSRContext** 関数に context 変数を渡すことによって、認証情報をリクエストに含めてくれる API オブジェクトを取得できる

https://aws.amazon.com/blogs/mobile/ssr-support-for-aws-amplify-javascript-libraries/

aws

Amplify JavaScript が SSR をサポート (3)

Amplify JS
v3.1.0 (9/3)

- GitHub Repository: https://github.com/aws-amplify/amplify-flutter
- Blog: https://aws.amazon.com/blogs/mobile/ssr-support-for-aws-amplify-javascript-libraries/
- Document: https://docs.amplify.aws/lib/ssr/q/platform/js
- Workshop: https://github.com/dabit3/next.js-amplify-workshop
- Deploy (Serverless Framework): https://github.com/serverless-nextjs/serverless-next.js
- Deploy (Amplify Console) PFR: https://github.com/aws-amplify/amplify-console/issues/412

aws

Amplify iOS/Android Updates

Amplify iOS が Swift Combine を サポート

Amplify iOS
v1.1 (8/12)

```swift
@State var getPostsToken: AnyCancellable?
func getPosts() {
    // 1
    getPostsToken = Amplify.DataStore.query(Post.self)
        .receive(on: DispatchQueue.main)
        .sink(
            receiveCompletion: { completion in
                switch completion {
                case .failure(let error):
                    // handle error
                    break

                case .finished:
                    // handle completed stream
                    break
                }
            },

            // 2
            receiveValue: { posts in
                // populate UI with posts
            }
        )
}
```

- Amplify iOS 1.1 以降で全ての非同期 API が Swift Combine をサポート
- 以前は非同期呼び出しに標準コールバックのみをサポートしていたがコールバック地獄になりがち
- Combine をサポートしたことでリアクティブプログラミングが可能に

https://aws.amazon.com/blogs/mobile/using-swift-combine-with-aws-amplify/

aws

Amplify Android が RxJava を サポート

Amplify Android v1.3.0 (9/1)

```java
// Time t1
RxAmplify.DataStore.save(blogPost).subscribe(
    // Time t3
    saveInfo -> Log.i(TAG, "Saved a blog post.", saveInfo),
    failure -> Log.e(TAG, "Failed to save.", failure)
);
// Time t2
```

- Amplify Android 1.3.0 以降で全ての非同期 API がRxJavaをサポート
- 以前は非同期呼び出しに標準コールバックのみをサポートしていたがコールバック地獄になりがち
- RxJava をサポートしたことでリアクティブプログラミングが可能に

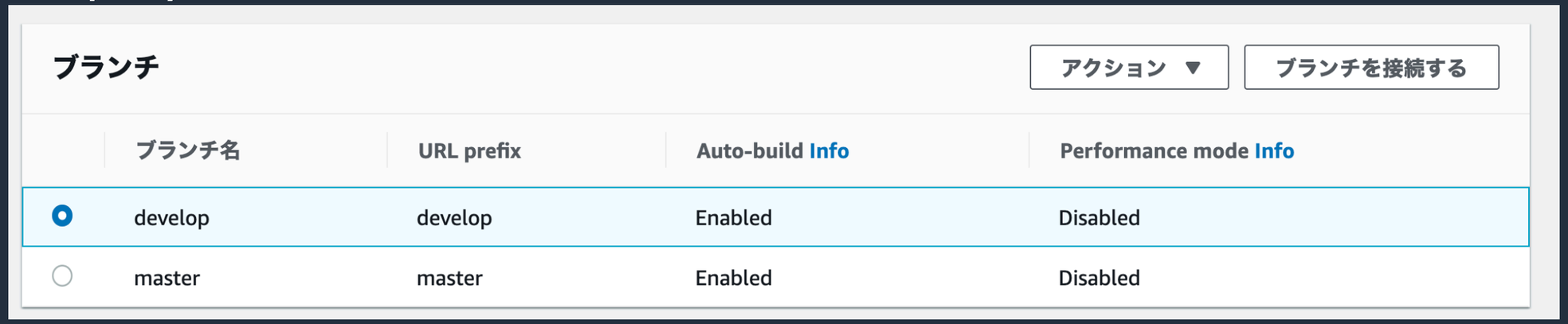
https://aws.amazon.com/blogs/mobile/using-rxjava-with-aws-amplify-android-library/

aws

Amplify Console Updates

Amplify Console - Perfomance Mode

Amplify Console > アプリの設定 > 全般

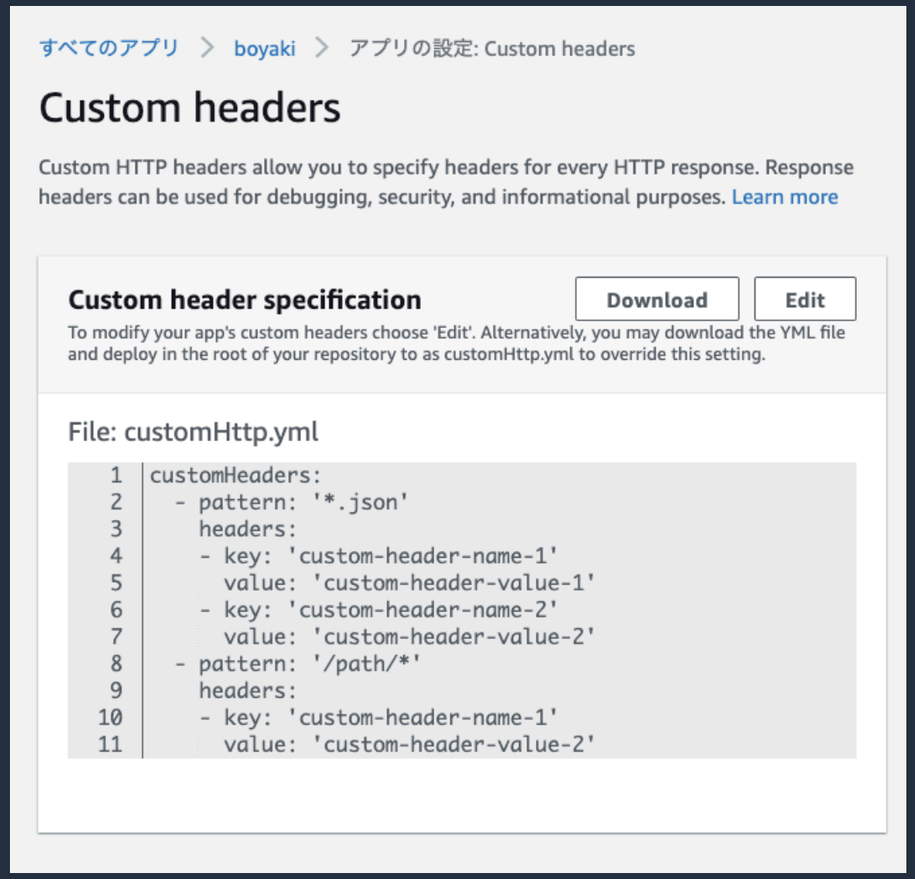


- CDNのエッジキャッシュがより長い時間保存され、ホスティング応答のパフォーマンスが向上する
- コードの変更が反映されるまで10分程度の時間を要するようになる

デプロイが即時的である必要がなく、応答速度が重要なワークロードに最適

https://docs.aws.amazon.com/ja_jp/amplify/latest/userguide/ttl.html#Performance-mode

Amplify Console – Custom Header (1)



- Custom Header を設定することで HTTP レスポンスに任意の HTTP ヘッダをつけることが可能
- 以前は コンソール上の “build設定の追加” やソースコード内の “amplify.yml” により指定する形式だったが、コンソール上の “Custome Headers” とソースコード内の customHttp.ymlが推奨に

https://docs.aws.amazon.com/ja_jp/amplify/latest/userguide/custom-headers.html

Amplify Console – Custom Header (2)


```yaml
customHeaders:
  - pattern: '**/*'
    headers:
      - key: 'Strict-Transport-Security'
        value: 'max-age=31536000; includeSubDomains'
      - key: 'X-Frame-Options'
        value: 'SAMEORIGIN'
      - key: 'X-XSS-Protection'
        value: '1; mode=block'
      - key: 'X-Content-Type-Options'
        value: 'nosniff'
      - key: 'Content-Security-Policy'
        value: 'default-src self'
```

- Custom HeaderによってHTTPSの強制、クロスサイトスクリプティング (XSS)攻撃の防止、クリックジャッキングの対策が可能
- ドキュメントに上記攻撃の対策を行うためのサンプルが記載

https://docs.aws.amazon.com/ja_jp/amplify/latest/userguide/custom-headers.html#example-security-headers

Thank you!