# Contributing Guidelines

Thank you for your interest in contributing to our project! <3 Whether it's a bug report, new feature, correction, or additional documentation, we greatly value feedback and contributions from our community. Please read through these guidelines carefully before submitting a PR or issue and let us know if it's not up-to-date (or even better, submit a PR with your corrections ;)).

- [Our History and Ethos](#our-history-and-ethos)

- [Our Design](#our-design)

- [Development Process](#development-process)

- [Setting up for local development](#setting-up-for-local-development)

- [Architecture of the codebase](#architecture-of-the-codebase)

- [Steps towards contributions](#steps-towards-contributions)

- [Pull Requests](#pull-requests)

- [Debugging](#debugging)

- [Release](#release)

- [Finding contributions to work on](#finding-contributions-to-work-on)

- [Related Repositories](#related-repositories)

- [Code of Conduct](#code-of-conduct)

- [Security issue notifications](#security-issue-notifications)

- [Licensing](#licensing)

# Our History and Ethos

AWS Amplify aims to enhance the development experience using JavaScript with AWS. Amplify codifies best practices through programmatic interfaces to help you effortlessly interact with cloud resources.

First and foremost Amplify exposes to you WHAT things do and then HOW best to do them. The WHAT is at a functional use case with HOW being an opinionated implementation that you can override with “escape hatches.” This will allow you to have higher velocity and build better applications by focusing less on implementation choices. Secondly, Amplify should be a manifestation of The Rule of Least Power when developing against AWS. This means it encourages architectural and programmatic best practices and the ability to start quickly. This shows by encouraging certain services (API Gateway usage vs. direct DynamoDB interaction) or certain connection patterns (Circuit breaker, retry counts and throttle up/down).

Opinionated implementations: There are many ways to interface with AWS Services. Certain service interactions are favored over others. For instance, if sending and receiving JSON, we would prefer an API Gateway endpoint to other mechanisms. Amplify will programmatically help optimize for cost and performance through library decisions.

Declarative actions: Amplify will provide you a reference to a generic client object and the ability to perform common actions. “RegisterUser”, “Login”, “SendObject”, “UpdateObject”, “StreamData”. By default you should not need to worry about AWS Service specific API operations like putItem() with a unique hash or even HTTP verbs.

Cascading service interactions: Certain actions in a declarative style can have overlapping or ambiguous AWS Service implementations. With an opinionated implementation, we can decide which Services are "primary" and which are "secondary" depending on what is configured. For instance, sending an image will prefer S3 over API Gateway.

Simple, standard data objects: Sending & receiving data to AWS Services can have many parameters, which tend to show up in the SDKs. These are abstracted and inferred, where possible, with simple JSON that the implementation can reason about. Standard parameters (bucket names, stream names, partition keys, etc.) that are part of the implementation are extracted from a simplified configuration file and dynamically generated/updated in order to further allow focus on state and data types only.

# Our Design

As more and more modules were introduced to AWS Amplify, it became necessary to modularize the library into smaller pieces so that users could avoid importing unnecessary parts into their app. The goal of this design is to make AWS Amplify modularized and also keep it backward-compatible to avoid breaking changes.

Modular import prevents unnecessary code dependencies from being included with the app, and thus decreases the bundle size while enabling added new functionality without the risk of introducing errors related to unused code.

Amplify has established the concepts of categories and plugins. A category is a collection of api calls that are exposed to the client to do things inside that category. For example, in the storage category, generally one wants to upload and download objects from storage so the apis exposed to the client will represent that functionality. Because Amplify is pluggable, a plugin of your choosing will provide the actual implementation behind that api interface. Using the same example of Storage, the plugin we choose might be AWSStoragePlugin which would then implement each api call from the category with a service call or set of service calls to S3, the underlying storage provider of the AWS plugin.

# Development Process

Our work is done directly on Github and PR's are sent to the github repo by core team members and contributors. Everyone undergoes the same review process to get their changes into the repo.

## Setting up for local development

This section should get you running with \*\*Amplify JS\*\* and get you familiar with the basics of the codebase. You will need the latest version of [nodejs](https://nodejs.org/en/) on your system and developing locally also requires `yarn` workspaces. You can install it [here](https://classic.yarnpkg.com/en/docs/install#mac-stable).

Start by, [Forking](https://help.github.com/en/github/getting-started-with-github/fork-a-repo) the main branch of [amplify-js](https://github.com/aws-amplify/amplify-js).

```

$ git clone git@github.com:[username]/amplify-js.git

$ cd amplify-js

$ yarn

$ yarn bootstrap

$ yarn build

```

> Note: Make sure to always [sync your fork](https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/syncing-a-fork) with main branch of amplify-js

## Architecture of the codebase

Amplify JS is a monorepo built with `Yarn` and `Lerna`. All the categories of Amplify live within the `packages` directory in the root. Each category inside packages has its own `src/` and `package.json`.

#### Packages inside Amplify JS

- [core](https://github.com/aws-amplify/amplify-js/tree/main/packages/core)

- [cache](https://github.com/aws-amplify/amplify-js/tree/main/packages/cache)

- [auth](https://github.com/aws-amplify/amplify-js/tree/main/packages/auth)

- [api](https://github.com/aws-amplify/amplify-js/tree/main/packages/api)

- [analytics](https://github.com/aws-amplify/amplify-js/tree/main/packages/analytics)

- [storage](https://github.com/aws-amplify/amplify-js/tree/main/packages/storage)

- [interactions](https://github.com/aws-amplify/amplify-js/tree/main/packages/interactions)

- [predictions](https://github.com/aws-amplify/amplify-js/tree/main/packages/predictions)

- [pubsub](https://github.com/aws-amplify/amplify-js/tree/main/packages/pubsub)

- [datastore](https://github.com/aws-amplify/amplify-js/tree/main/packages/datastore)

- [pushnotification](https://github.com/aws-amplify/amplify-js/tree/main/packages/pushnotification)

- [xr](https://github.com/aws-amplify/amplify-js/tree/main/packages/xr)

- [amplify-ui](https://github.com/aws-amplify/amplify-js/tree/main/packages/amplify-ui)

- [aws-amplify-react](https://github.com/aws-amplify/amplify-js/tree/main/packages/aws-amplify-react)

- [aws-amplify-react-native](https://github.com/aws-amplify/amplify-js/tree/main/packages/aws-amplify-react-native)

- [aws-amplify-angular](https://github.com/aws-amplify/amplify-js/tree/main/packages/aws-amplify-angular)

- [aws-amplify-vue](https://github.com/aws-amplify/amplify-js/tree/main/packages/aws-amplify-vue)

- [amplify-ui-components](https://github.com/aws-amplify/amplify-js/tree/main/packages/amplify-ui-components)

- [amplify-ui-react](https://github.com/aws-amplify/amplify-js/tree/main/packages/amplify-ui-react)

- [amplify-ui-angular](https://github.com/aws-amplify/amplify-js/tree/main/packages/amplify-ui-angular)

- [amplify-ui-vue](https://github.com/aws-amplify/amplify-js/tree/main/packages/amplify-ui-vue)

- [amplify-ui-storybook](https://github.com/aws-amplify/amplify-js/tree/main/packages/amplify-ui-storybook)

## Steps towards contributions

- To make changes with respect to a specific category, go into `packages/[category]`.

- Make changes to required file.

- Write unit tests

- Yarn build

- Run test suite

- Test in sample app using yarn linking

- Submit a PR

#### Build step:

```

$ yarn build --scope @aws-amplify/auth

```

#### Testing:

```

$ yarn run test --scope @aws-amplify/auth

$ yarn run test --scope @aws-amplify/ui-components

```

> Note: There is a commit hook that will run the tests prior to committing. Please make sure if you are going to provide a pull request to be sure you include unit tests with your functionality and that all tests pass.

#### Test in a local sample app

\*\*Yarn Linking\*\*

The best way to develop locally and test is to link the individual package you’re working on and run lerna in watch mode.

Run watch mode while editing (auth for example):

```

$ npx lerna exec --scope @aws-amplify/auth yarn link

$ npx lerna exec --scope @aws-amplify/auth yarn build:esm:watch

```

Or run the whole library in watch mode if you are working on multiple packages

```

$ yarn build # Build the whole library

$ yarn link-all # Make all the packages available to link

$ yarn build:esm:watch # All packages are building ES6 modules in watch mode

```

In you sample project, you can now link specific packages

```

$ yarn link @aws-amplify/auth

```

These tests are only necessary if you’re looking to contribute a Pull Request. If you’re just playing locally you don’t need them. However if you’re contributing a Pull Request for anything other than bug fixes it would be best to validate that first because depending on the scope of the change.

#### Verdaccio

Verdaccio is a lightweight private npm proxy registry built in Node.js. Install [Verdaccio](https://verdaccio.org/docs/en/installation). You can setup veradaccio to publish packages locally to test the changes.

To publish in Verdaccio, start a verdaccio instance and then,

```

npm set registry http://localhost:4873/

yarn lerna publish --skip-git --force-publish

```

To publish a local version of a specific package,

```

cd packages/<category>

npm publish --registry http://localhost:4873 (http://localhost:4873/)

```

Once you are done with Verdaccio, you can reset to npm registry by doing,

```

npm set registry https://registry.npmjs.com/

```

# Pull Requests

This is mostly the same as GitHub's guide on creating a pull request.

\_[Skip step 1 to 3 if you have already done this]\_

1. Fork aws-amplify/amplify-js

2. Clone your fork locally: `git clone git@github.com:YOUR\_GITHUB\_USERNAME/amplify-js.git`

3. Run `yarn && yarn bootstrap` in the repository root

4. Within your fork, create a new branch based on the issue (e.g. Issue #123) you're addressing - `git checkout -b "group-token/short-token-[branch-name]"` or `git checkout -b "short-token/[branch-name]"`

- Use grouping tokens at the beginning of the branch names. \\_For e.g, if you are working on changes specific to `amplify-ui-components`, then you could start the branch name as `ui-components/...`

- short token

- feat

- chore

- bug

- use slashes to seperate parts of branch names

- Hyphenate well defined branch name

5. Once your work is committed and you're ready to share, run test `yarn test`.

\*\*Note:\*\* Manually test your changes in a sample app with different edge cases and also test across different browsers and platform

6. Then, Push your branch `git push origin -u`

7. This previous step will give you a URL to view a GitHub page in your browser. Copy-paste this, and complete the workflow in the UI. It will invite you to "create a PR" from your newly published branch. Fill out the PR template to submit a PR.

8. Finally, the Amplify JS team will review your PR. Add reviewers based on the core member who is tracking the issue with you or code owners.

\_In the meantime, address any automated check that fail (such as linting, unit tests, etc. in CI)\_

# Debugging

Sometimes the issue can be solved by doing a clean and fresh build. To do this, make sure to remove your node modules and clean your packages. You can run `git clean -xdf` in the repository root to achieve this.

# Release

To give a bird's eye view of the release cycle,

- We follow semantic versioning for our releases

- Every merge into the `main` ends up as `unstable` package in the npm

- The core team will cut a release out to `stable` from `unstable` bi-weekly

## Finding contributions to work on

Looking at the existing issues is a great way to find something to contribute on. As our projects, by default, use the default GitHub issue labels (enhancement/bug/duplicate/help wanted/invalid/question/wontfix), looking at any ['help wanted'](https://github.com/aws-amplify/amplify-ios/labels/help%20wanted) or ['good first issue'](https://github.com/aws-amplify/amplify-ios/issues?q=is%3Aopen+is%3Aissue+label%3A%22good+first+issue%22) is a great place to start.

You could also contribute by reporting bugs, reproduction of bugs with sample code, documentation and test improvements.

## Related Repositories

The Amplify Framework runs on Android, iOS, and numerous JavaScript-based web platforms.

1. [AWS Amplify for iOS](https://github.com/aws-amplify/amplify-ios)

2. [AWS Amplify for Android](https://github.com/aws-amplify/amplify-android)

3. [AWS Amplify CLI](https://github.com/aws-amplify/amplify-cli)

AWS Amplify plugins are built on top of the AWS SDKs. AWS SDKs are a

toolkit for interacting with AWS backend resources.

1. [AWS SDK for Android](https://github.com/aws-amplify/aws-sdk-android)

2. [AWS SDK for iOS](https://github.com/aws-amplify/aws-sdk-ios)

3. [AWS SDK for JavaScript](https://github.com/aws/aws-sdk-js)

## Code of Conduct

This project has adopted the [Amazon Open Source Code of Conduct](https://aws.github.io/code-of-conduct).

For more information see the [Code of Conduct FAQ](https://aws.github.io/code-of-conduct-faq) or contact

opensource-codeofconduct@amazon.com with any additional questions or comments.

## Security issue notifications

If you discover a potential security issue in this project we ask that you notify AWS/Amazon Security via our [vulnerability reporting page](http://aws.amazon.com/security/vulnerability-reporting/). Please do \*\*not\*\* create a public github issue.

## Licensing

See the [LICENSE](https://github.com/aws-amplify/amplify-js/blob/main/LICENSE) file for our project's licensing. We will ask you to confirm the licensing of your contribution.

We may ask you to sign a [Contributor License Agreement (CLA)](http://en.wikipedia.org/wiki/Contributor\_License\_Agreement) for larger changes.