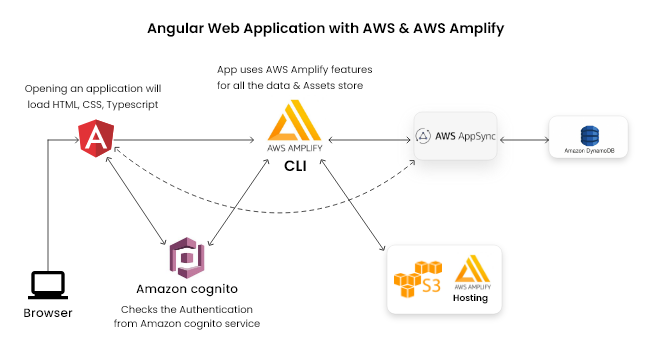
Angular is a platform and framework for building SPAs or Single-Page Applications using TypeScript and HTML. However, AWS, or Amazon Web Services, is one of the most prominent cloud platforms that present numerous feature-packed services from data centers around the globe.

AWS presents a feature called AWS Amplify that allows you to quickly choose the desired development platform that is your ideal choice. This saves time and ensures the quality of the front end, which is even more complimented by the **Angular with AWS combination** that allows you to have high-quality, user-friendly applications that are effective, efficient, and highly scalable. Now, before jumping into the details of the topic, let us have an overview of the relationship between the Angular AWS combination.

**Relationship between Angular with AWS**

The relationship between Angular with AWS can be quickly and broadly understood with the architecture that explains the functioning of the two. Below is the Angular application architecture configured with AWS Amplify using Amplify CLI. This Amplify CLI allows adding features like Amazon DynamoDB, AWS Cognito, AWS S3 Storage, Amplify Hosting, and others.



* Looking at the architecture, we realize that the client can access the web application built on Angular at the front end and AWS at the back end.
* For accessing the AWS backend in the client application, it is crucial to use the AWS Amplify toolchain to connect the server and the client.
* Using the AWS Amplify CLI, we add the authentication powered by Amazon Cognito.
* Then, using the same AWS CLI, we added DataStore powered by AWS DynamDB powered by AWS AppSync, again using the Amplify CLI.
* Then, with the same AWS Amplify CLI, we stored the application assets in AWS S3 storage.
* Also, we have used the AWS Amplify hosting for our site on AWS S3 storage.

**What is (AWS) Amazon Web Services?**

Amazon Web Services (AWS) is the world’s most comprehensively evolving cloud computing platform, launched in 2002 by Amazon, that offers more than 200 featured services from data centers around the globe. AWS is one of the most extensively adopted cloud platforms that present a number of on-demand operations, like content delivery, compute power, database storage, and others, to help you scale and grow.

Features of AWS or Amazon Web Services

Millions of customers, including fast-growing start-ups, big companies, and even leading government agencies, use AWS to lower their investment costs, become more agile, and innovate faster. The reason behind this extensive popularity lies within the features of AWS. To get a bit more clarification on this, let us have a look at the features of AWS:

* Mobile Friendly Access
* Serverless Cloud Functions
* Databases
* Storage
* Security and Compliance

Other Features of AWS:

* Cost Effective
* Experienced
* Scalable and Elastic
* Flexibility

**What is AWS Amplify?**

AWS Amplify helps developers to build and deploy serverless applications efficiently in the AWS cloud. It is a full-stack application platform that provides many features like Authentication, Database, Storage, Hosting, and many more.

The most enticing factor about AWS Amplify is that it allows the full-stack building of web and mobile apps in hours that are easy to start and scale. It creates a cross-platform backend for your web or Mobile Application with the features of authentication, data, storage, and many more in minutes.

It gives the ability to build frontend UI from prototype to code through Figma integration. It connects the App front-end with the Back-end backed by AWS and, Deploys the applications fast, securely, and reliably in clicks.

The features of AWS Amplify include the following:

* Authentication
* Analytics
* DataStore
* API
* Functions
* Geo
* Interactions
* Predictions
* PubSub
* Push Notifications
* Storage

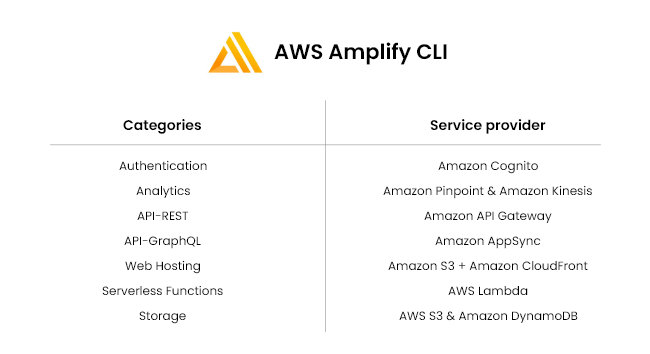
**Why use AWS Amplify with Angular?**

The AWS Amplify offers a number of services, such as:

* **Amplify CLI**, which helps calibrate the services that assist in powering the backend via a simple command line interface.
* **Amplify Libraries** allows you to use case-entric client libraries to integrate your application code using declarative interfaces to the backend.
* **Amplify UI Components** which offers the UI libraries for frameworks like React Native, React, Angular, Vue, and Flutter.
* **Amplify Hosting**is one of the core AWS services that offer a git-based workflow for the continuous deployment and hosting of full-stack web applications. You can also find the cloud resources created by the Amplify CLI in the Amplify Console.

**What is Amplify CLI, and Why use it?**

In simple words, we can say that the Amplify CLI acts as a unified toolchain for creating, integrating, and managing the AWS cloud services for your applications. It complements the AWS Amplify Library to offer you end-to-end solutions for creating, configuring, and implementing scalable cloud-based applications along with a Rails-like experience focusing on the developer’s efficiency. AWS Amplify supports not only JavaScript applications, frameworks, or languages such as Angular, React, Vue, Swift (Native iOS), Android (Native Android), and Flutter.



**Deploying Angular Application with AWS?**

Suppose you wish to deploy your Angular application with AWS infrastructure. In that case, there are two options that you can go with to perform the same:

* Static web app hosting in AWS S3 storage

Automatic deployment from

* github/Bitbucket/Gitlab repository. (Need to add Amplify hosting from the Amplify CLI)

For the Static Way of deployment, the application file is a manual process where you must upload all your application build bundle files. In the Second approach, with dynamic deployment from the github repository branches, you must manually configure the build script. The script will deploy the application automatically based on your written script.

# Prerequisites

Before you begin, make sure you have the following installed:

* [Node.js](https://nodejs.org/) v14.x or later
* [npm](https://www.npmjs.com/) v6.14.4 or later
* [git](https://git-scm.com/) v2.14.1 or later

## [Sign up for an AWS account](https://docs.amplify.aws/start/getting-started/installation/q/integration/angular/" \l "sign-up-for-an-aws-account)

If you don't already have an AWS account, you'll need to create one in order to follow the steps outlined in this tutorial.

[Create AWS Account](https://portal.aws.amazon.com/billing/signup?redirect_url=https%3A%2F%2Faws.amazon.com%2Fregistration-confirmation#/start)

There are no upfront charges or any term commitments to create an AWS account and signing up gives you immediate access to the AWS Free Tier.

## [Configuring the Amplify CLI](https://docs.amplify.aws/start/getting-started/installation/q/integration/angular/" \l "configuring-the-amplify-cli)

To set up the Amplify CLI on your local machine, you have to configure it to connect to your AWS account.

If you already have an AWS profile with credentials on your machine, you can skip this step.

If you are using CLI version < 10.8.0 go to [Configure CLI < 10.8.0](https://docs.amplify.aws/start/getting-started/installation/q/integration/angular/#configure-the-amplify-cli-version--1080)

Configure Amplify by running the following command:

npm install -g @aws-amplify/cli

amplify configure

amplify configure will ask you to sign into the AWS Console.

Once you're signed in, Amplify CLI will ask you to create an IAM user.

Amazon IAM (Identity and Access Management) enables you to manage users and user permissions in AWS. You can learn more about Amazon IAM [here](https://aws.amazon.com/iam/).

Specify the AWS Region

? region: # Your preferred region

Follow the instructions at

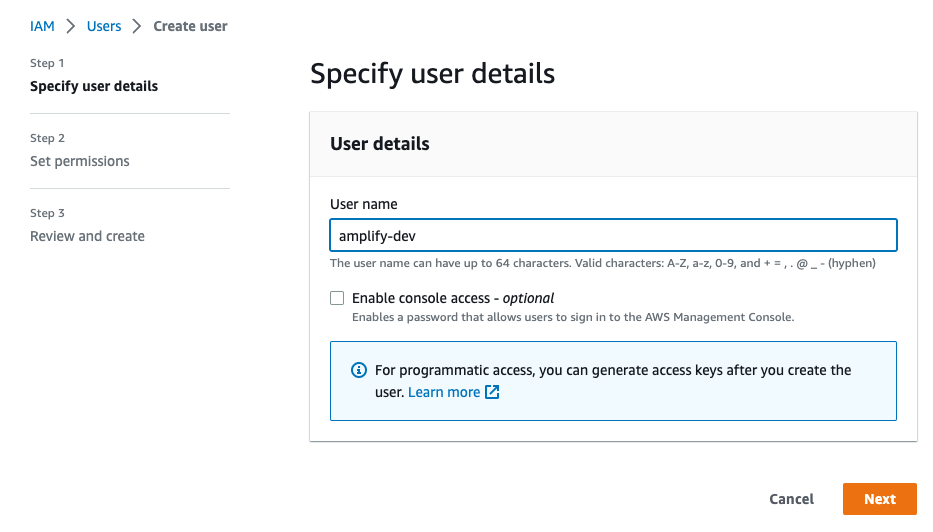
https://docs.amplify.aws/cli/start/install/#configure-the-amplify-cli

to complete the user creation in the AWS console

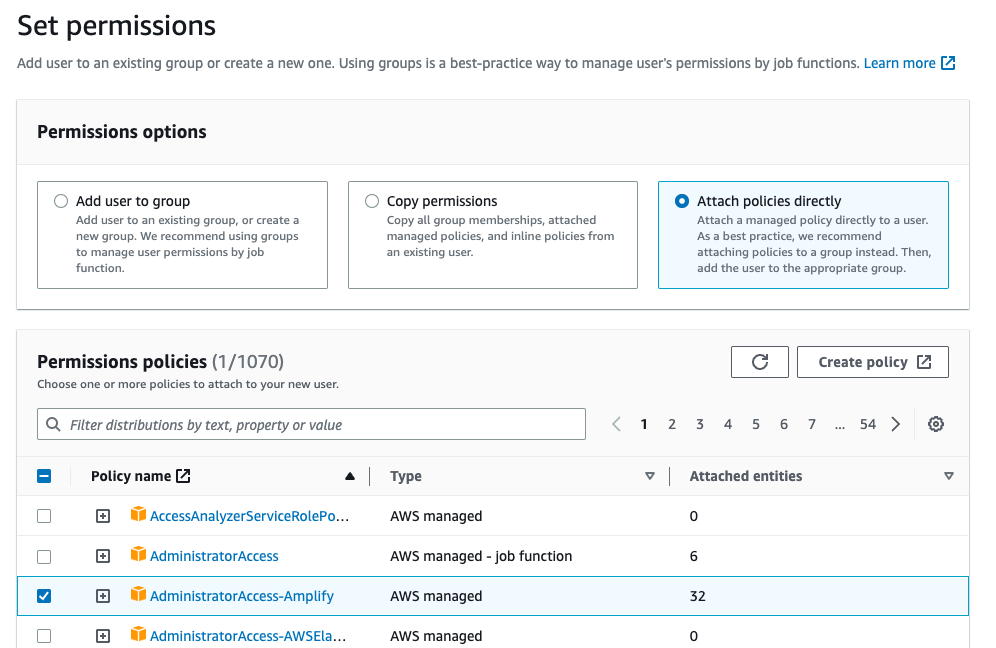
https://console.aws.amazon.com/iamv2/home#/users/create

Navigate to the [IAM User creation page](https://console.aws.amazon.com/iamv2/home#/users/create) if it's not already open.

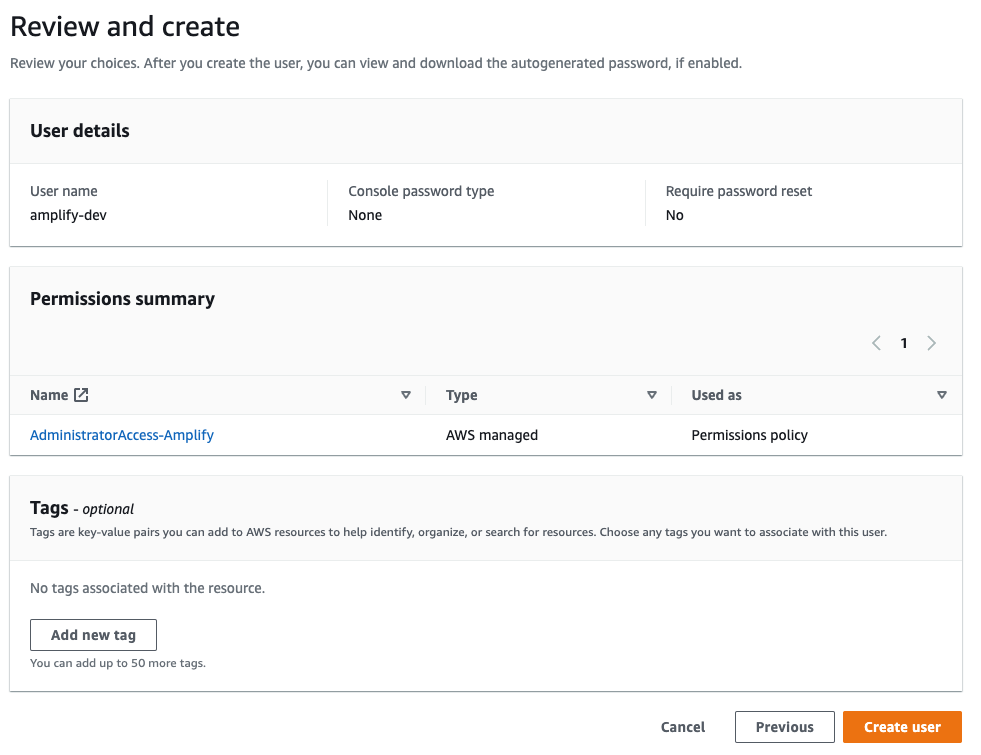
Enter a User name and select **Next**. You can name the user anything but we'll call it "amplify-dev".



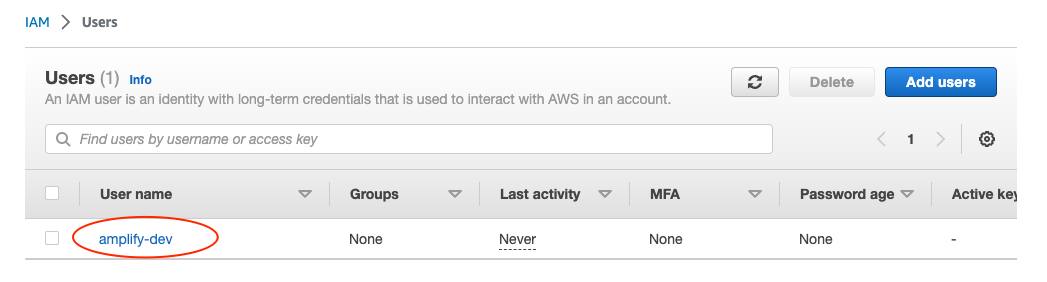
Select **Attach policies directly** and select **AdministratorAccess-Amplify** as the Permissions policy. Select **Next**.



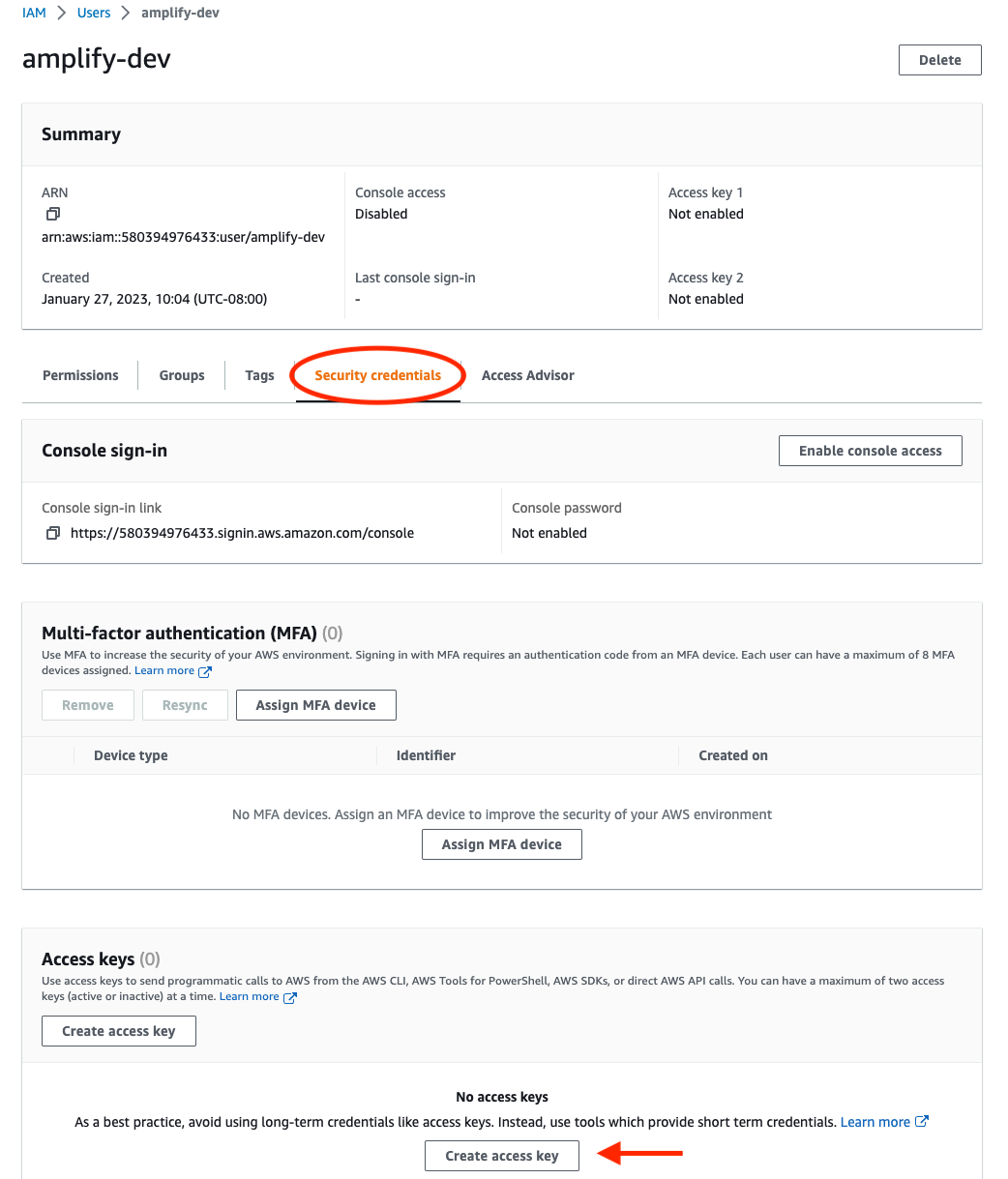
On the Review page, check that everything looks good and select **Create user**.



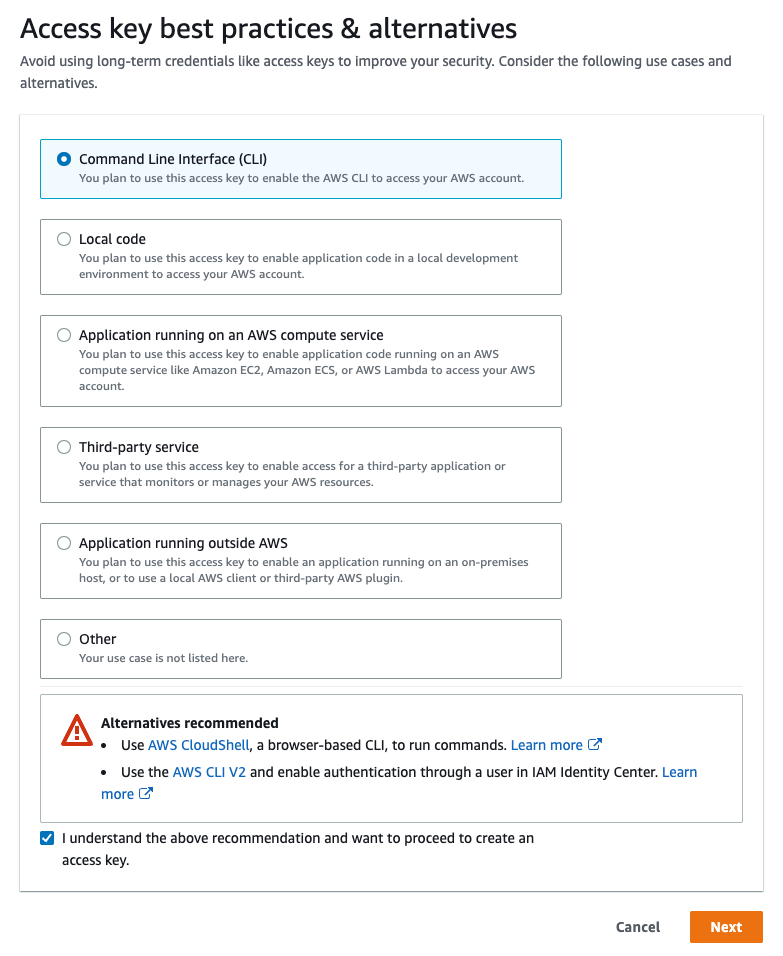
This will redirect to the users list page. Select the user you just created.



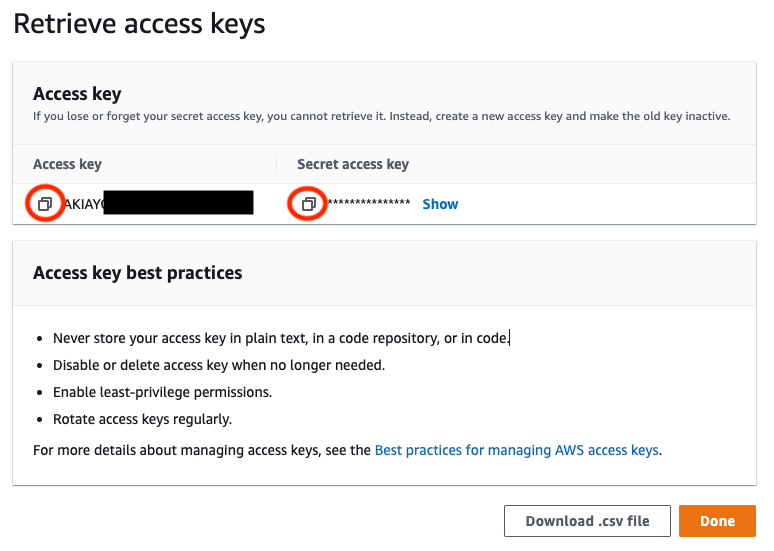
On the user details page, navigate to the **Security credentials** tab, scroll down to **Access keys** and select **Create access keys**.



On the next page, select **Command Line Interface**, acknowledge the warning, and select **Next**.



On the next page select **Create access key**. You’ll then see a page with the access keys for the user. Use the copy icon to copy these values to your clipboard, then return to the Amplify CLI.



Enter the values you just copied into the corresponding CLI prompts.

Enter the access key of the newly created user:

? accessKeyId: # YOUR\_ACCESS\_KEY\_ID

? secretAccessKey: # YOUR\_SECRET\_ACCESS\_KEY

This would update/create the AWS Profile in your local machine

? Profile Name: # (default)

Successfully set up the new user.

## [Configure the Amplify CLI version < 10.8.0](https://docs.amplify.aws/start/getting-started/installation/q/integration/angular/" \l "configure-the-amplify-cli-version--1080)

### [Option 1: Watch the video guide](https://docs.amplify.aws/start/getting-started/installation/q/integration/angular/" \l "option-1-watch-the-video-guide)

Watch the video below to learn how to install and configure the Amplify CLI or skip to the next section to follow the step-by-step instructions.

### [Option 2: Follow the instructions](https://docs.amplify.aws/start/getting-started/installation/q/integration/angular/" \l "option-2-follow-the-instructions)

Configure Amplify by running the following command:

amplify configure

copy

amplify configure will ask you to sign into the AWS Console.

Once you're signed in, Amplify CLI will ask you to create an IAM user.

Amazon IAM (Identity and Access Management) enables you to manage users and user permissions in AWS. You can learn more about Amazon IAM [here](https://aws.amazon.com/iam/).

Specify the AWS Region

? region: # Your preferred region

Specify the username of the new IAM user:

? user name: # User name for Amplify IAM user

Complete the user creation using the AWS console

Create a user with AdministratorAccess-Amplify to your account to provision AWS resources for you like AppSync, Cognito etc.

Once the user is created, Amplify CLI will ask you to provide the accessKeyId and the secretAccessKey to connect Amplify CLI with your newly created IAM user.

Enter the access key of the newly created user:

? accessKeyId: # YOUR\_ACCESS\_KEY\_ID

? secretAccessKey: # YOUR\_SECRET\_ACCESS\_KEY

This would update/create the AWS Profile in your local machine

? Profile Name: # (default)

Successfully set up the new user.

It is recommended to create a new IAM user for every device that installs the Amplify CLI, rather than attempt to use an existing IAM user used on another device. Having a distinct user for each machine provides the best level of visibility and control without breaking the deployment of your app, allowing for the quick deactivation of an individual machine if needed.

Successfully set up the new user.

# Set up fullstack project

Use the [Angular CLI](https://github.com/angular/angular-cli) to bootstrap a new Angular app:

123456

npx -p @angular/cli ng new amplify-app

? Would you like to add Angular routing? Y

? Which stylesheet format would you like to use? (your preferred stylesheet provider)

cd amplify-app

copy

### [Angular 6+ Support](https://docs.amplify.aws/start/getting-started/setup/q/integration/angular/" \l "angular-6-support)

Angular 6+ does not include shims for 'global' or 'process' as provided in previous versions. Add the following to your src/polyfills.ts file to recreate them:

1234

(window as any).global = window;

(window as any).process = {

env: { DEBUG: undefined },

};

copy

### [Internet Explorer 11 (IE11) Support:](https://docs.amplify.aws/start/getting-started/setup/q/integration/angular/" \l "internet-explorer-11-ie11-support)

In order for Angular apps to work on IE11, you need to add the following to your src/polyfills.ts file as well:

12

import 'core-js/es/typed-array';

import 'core-js/es/object';

copy

## [Create a new Amplify backend](https://docs.amplify.aws/start/getting-started/setup/q/integration/angular/" \l "create-a-new-amplify-backend)

Now that you have a running Angular app, it's time to set up Amplify for this app so that you can create the necessary backend services needed to support the app.

Open a new terminal. From the root of the project, run:

amplify init

copy

When you initialize Amplify you'll be prompted for some information about the app. For newer versions of Angular, you will have to change the Distribution Directory Path from dist to dist/amplify-app to match how Angular will build your project.

? Enter a name for the project: amplifyapp

The following configuration will be applied:

Project information

| Name: amplifyapp

| Environment: dev

| Default editor: Visual Studio Code

| App type: javascript

| Javascript framework: angular

| Source Directory Path: src

| Distribution Directory Path: dist

| Build Command: npm run-script build

| Start Command: ng serve

? Initialize the project with the above configuration? Yes

Using default provider awscloudformation

? Select the authentication method you want to use: AWS profile

For more information on AWS Profiles, see:

https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html

? Please choose the profile you want to use: default

When you initialize a new Amplify project, a few things happen:

* It creates a top level directory called amplify that stores your backend definition. During the tutorial you'll add capabilities such as authentication, GraphQL API, storage, and set up authorization rules for the API. As you add features, the amplify folder will grow with infrastructure-as-code templates that define your backend stack. Infrastructure-as-code is a best practice way to create a replicable backend stack.
* It creates a file called aws-exports.js in the src directory that holds all the configuration for the services you create with Amplify. This is how the Amplify client is able to get the necessary information about your backend services.
* It modifies the .gitignore file, adding some generated files to the ignore list.
* A cloud project is created for you in the AWS Amplify Console that can be accessed by running amplify console. The Console provides a list of backend environments, deep links to provisioned resources per Amplify category, status of recent deployments, and instructions on how to promote, clone, pull, and delete backend resources.

## [Install Amplify libraries](https://docs.amplify.aws/start/getting-started/setup/q/integration/angular/" \l "install-amplify-libraries)

Inside the amplify-app directory, install the Amplify Angular library and run your app:

123

npm install --save aws-amplify @aws-amplify/ui-angular

npm start

copy

The @aws-amplify/ui-angular package is a set of Angular components and an Angular provider which helps integrate your application with the AWS-Amplify library.

Angular CLI output warnings: if you see CommonJS or AMD dependencies optimization bailouts warnings using Angular 9+, you can use this [gist](https://gist.github.com/wlee221/6d98d96740bea6f53327b4db4a432616) to remove them. More details about these [here](https://angular.io/guide/build#configuring-commonjs-dependencies).

### [Strictly typing aws-exports](https://docs.amplify.aws/start/getting-started/setup/q/integration/angular/" \l "strictly-typing-aws-exports)

If you have TypeScript [strict mode](https://www.typescriptlang.org/tsconfig/#strict) on and see the error

Could not find a declaration file for module './aws-exports'. 'aws-exports.js' implicitly has an 'any' type.

copy

Create a aws-exports.d.ts file on the same level as aws-exports with the following content:

12

declare const awsmobile: Record<string, any>

export default awsmobile;

copy

### [Importing the Amplify Angular UI Module](https://docs.amplify.aws/start/getting-started/setup/q/integration/angular/" \l "importing-the-amplify-angular-ui-module)

Add the **Amplify Authenticator UI Module** to src/app/app.module.ts:

12345678910111213141516171819202122

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AmplifyAuthenticatorModule } from '@aws-amplify/ui-angular';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

AppRoutingModule,

/\* configure App with AmplifyAuthenticatorModule \*/

AmplifyAuthenticatorModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

# Connect API and database to the app

### [Model the data with GraphQL transform](https://docs.amplify.aws/start/getting-started/data-model/q/integration/angular/" \l "model-the-data-with-graphql-transform)

Add a [GraphQL API](https://docs.aws.amazon.com/appsync/latest/devguide/designing-a-graphql-api.html) to your app and automatically provision a database by running the following command from the root of your application directory:

amplify add api

copy

Accept the **default values** which are highlighted below:

? Select from one of the below mentioned services:

# GraphQL

? Here is the GraphQL API that we will create. Select a setting to edit

or continue

# Continue

? Choose a schema template:

# Single object with fields (e.g., “Todo” with

ID, name, description)

✔ Do you want to edit the schema now? (Y/n)

# yes

The generated schema is for a Todo app. Replace the GraphQL schema at amplify/backend/api/RestaurantAPI/schema.graphql with the following:

123456

type Restaurant @model {

id: ID!

name: String!

description: String!

city: String!

}

copy

You'll notice a directive on the Restaurant type of @model. This directive is part of Amplify's [GraphQL transformer](https://docs.amplify.aws/cli/graphql/data-modeling/) functionality.

The GraphQL Transform Library provides custom directives you can use in your schema that allow you to do things like define data models, set up authentication and authorization rules, configure serverless functions as resolvers, and more.

A type decorated with the @model directive will scaffold out the database table for the type (Restaurant table), the schema for CRUD (create, read, update, delete) and list operations, and the GraphQL resolvers needed to make everything work together.

From the command line, press **enter** to accept the schema and continue to the next steps.

## [Creating the API with database](https://docs.amplify.aws/start/getting-started/data-model/q/integration/angular/" \l "creating-the-api-with-database)

Create required backend resources for your configured api with the following command:

123456

amplify push

? Are you sure you want to continue? Y

? Do you want to update code for your updated GraphQL API Yes

? Do you want to generate GraphQL statements (queries, mutations and subscription) based on your schema types?

This will overwrite your current graphql queries, mutations and subscriptions: Yes

copy

### [Code generation](https://docs.amplify.aws/start/getting-started/data-model/q/integration/angular/" \l "code-generation)

Once the deployment is complete, the CLI will create a new directory in src/graphql with all of the GraphQL operations you will need for your API. The CLI also created an API.service.ts file in the app directory that you will be using shortly.

Next, run the following command to check Amplify's status:

amplify status

copy

This will give us the current status of the Amplify project, including the current environment, any categories that have been created, and what state those categories are in. It should look similar to this:

Current Environment: dev

| Category | Resource name | Operation | Provider plugin |

| -------- | ------------- | --------- | ----------------- |

| Api | RestaurantAPI | No Change | awscloudformation |

### [Testing your API](https://docs.amplify.aws/start/getting-started/data-model/q/integration/angular/" \l "testing-your-api)

You can open the AWS console to run Queries, Mutation, or Subscription against your new API at any time directly by running the following command:

amplify console api

copy

When prompted, select **GraphQL**. This will open the AWS AppSync console for you to run Queries, Mutations, or Subscriptions at the server and see the changes in your client app.

## [Connect frontend to API](https://docs.amplify.aws/start/getting-started/data-model/q/integration/angular/" \l "connect-frontend-to-api)

Open src/main.ts and add the following code to configure the Angular project with Amplify:

import { Amplify } from 'aws-amplify';

import aws\_exports from './aws-exports';

Amplify.configure(aws\_exports);

copy

On Angular 12+, Typescript [strict mode](https://www.typescriptlang.org/tsconfig#strict) is enabled by default, which will introduce some type errors when you run the application. Please see the TypeScript strict mode [troubleshooting guide](https://docs.amplify.aws/lib/troubleshooting/strict-mode/q/platform/js/) to resolve them.

Update tsconfig.app.json to include the "node" compiler option in types:

123

"compilerOptions": {

"types" : ["node"]

}

copy

Depending on your TypeScript version you may need to rename aws-exports.js to aws-exports.ts prior to importing, or enable the allowJs [compiler option](https://www.typescriptlang.org/docs/handbook/compiler-options.html) in your tsconfig.

First, enable the Angular forms modules in src/app/app.module.ts:

12345678910111213141516171819202122232425

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AmplifyAuthenticatorModule } from '@aws-amplify/ui-angular';

/\* new form imports \*/

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { RestaurantsComponent } from './restaurants/restaurants.component';

@NgModule({

declarations: [AppComponent, RestaurantsComponent],

imports: [

BrowserModule,

AppRoutingModule,

AmplifyAuthenticatorModule,

/\* configuring form modules \*/

FormsModule,

ReactiveFormsModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule {}

copy

Next, create an restaurants Angular component:

npx ng generate component restaurants

copy

Open src/app/app.component.html, and replace the default content with the Angular component you created:

<app-restaurants></app-restaurants>

copy

In your src/app/restaurants/restaurants.component.ts file, add data to your database with a mutation by using the API.service file which was generated when you ran amplify add api:

1234567891011121314151617181920212223242526272829303132

import { Component } from '@angular/core';

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

import { APIService, Restaurant } from '../API.service';

@Component({

selector: 'app-restaurants',

templateUrl: './restaurants.component.html',

styleUrls: ['./restaurants.component.css']

})

export class RestaurantsComponent {

public createForm: FormGroup;

constructor(private api: APIService, private fb: FormBuilder) {

this.createForm = this.fb.group({

name: ['', Validators.required],

description: ['', Validators.required],

city: ['', Validators.required]

});

}

public onCreate(restaurant: Restaurant) {

this.api

.CreateRestaurant(restaurant)

.then(() => {

console.log('item created!');

this.createForm.reset();

})

.catch((e) => {

console.log('error creating restaurant...', e);

});

}

}

copy

Next, add a form that will be used for creating restaurants. Add the following to your src/app/restaurants/restaurants.component.html:

123456789101112131415161718192021

<div class="form-body">

<form

autocomplete="off"

[formGroup]="createForm"

(ngSubmit)="onCreate(createForm.value)"

>

<div>

<label>Name: </label>

<input type="text" formControlName="name" autocomplete="off" />

</div>

<div>

<label>Description: </label>

<input type="text" formControlName="description" autocomplete="off" />

</div>

<div>

<label>City: </label>

<input type="text" formControlName="city" autocomplete="off" />

</div>

<button type="submit">Submit</button>

</form>

</div>

copy

Next, update your RestaurantsComponent class so that it will list all restaurants in the database when the app starts. To do so, implement [OnInit](https://angular.io/api/core/OnInit) add a ListRestaurants query in src/app/restaurants/restaurants.component.ts. Store the query results in an array.

123456789101112131415161718192021222324252627282930313233343536373839404142

import { Component, OnInit } from '@angular/core';

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

import { APIService, Restaurant } from '../API.service';

@Component({

selector: 'app-restaurants',

templateUrl: './restaurants.component.html',

styleUrls: ['./restaurants.component.css']

})

export class RestaurantsComponent implements OnInit {

public createForm: FormGroup;

/\* declare restaurants variable \*/

public restaurants: Array<Restaurant> = [];

constructor(private api: APIService, private fb: FormBuilder) {

this.createForm = this.fb.group({

name: ['', Validators.required],

description: ['', Validators.required],

city: ['', Validators.required]

});

}

async ngOnInit() {

/\* fetch restaurants when app loads \*/

this.api.ListRestaurants().then((event) => {

this.restaurants = event.items as Restaurant[];

});

}

public onCreate(restaurant: Restaurant) {

this.api

.CreateRestaurant(restaurant)

.then((event) => {

console.log('item created!');

this.createForm.reset();

})

.catch((e) => {

console.log('error creating restaurant...', e);

});

}

}

copy

Add the following to your src/app/restaurants/restaurants.component.html to display any of the restaurants you have added:

12345

<div \*ngFor="let restaurant of restaurants">

<div>{{ restaurant?.city }}</div>

<div>{{ restaurant?.name }}</div>

<div>{{ restaurant?.description }}</div>

</div>

copy

To subscribe to realtime data, declare a subscription class variable and update ngOnInit in src/app/app.component.ts. When the app starts, this code will set up a subscription. The subscription will update the restaurants array when new events are received (when a new restaurant is created):

123456789101112131415161718192021

/\*\* Subscription type will be inferred from this library \*/

import { ZenObservable } from 'zen-observable-ts';

// ...

/\*\* Declare subscription variable \*/

private subscription: ZenObservable.Subscription | null = null;

// ...

async ngOnInit() {

this.api.ListRestaurants().then(event => {

this.restaurants = event.items as Restaurant[];

});

/\* subscribe to new restaurants being created \*/

this.subscription = this.api.OnCreateRestaurantListener().subscribe(

(event: any) => {

const newRestaurant = event.value.data.onCreateRestaurant;

this.restaurants = [newRestaurant, ...this.restaurants];

}

);

}

Finally, unsubscribe from the subscription when the component is destroyed. Import and add OnDestroy in src/app/app.component.ts:

12345678910

import { Component, OnDestroy, OnInit } from '@angular/core';

export class AppComponent implements OnInit, OnDestroy {

// ...

ngOnDestroy() {

if (this.subscription) {

this.subscription.unsubscribe();

}

this.subscription = null;

}

copy

Next, run the app:

npm start

copy

Now, open the app in 2 browser windows (both at <http://localhost:4200/>) so that you have your app running side by side. When creating a new item in one window, you should see it come through in the other window in real-time.

You have successfully deployed your API and connected it to your app!

# Add authentication

The next feature you will be adding is authentication.

## [Authentication with Amplify](https://docs.amplify.aws/start/getting-started/auth/q/integration/angular/" \l "authentication-with-amplify)

Amplify uses [Amazon Cognito](https://aws.amazon.com/cognito/) as the main authentication provider. Amazon Cognito is a robust user directory service that handles user registration, authentication, account recovery & other operations. In this tutorial, you'll learn how to add authentication to your application using Amazon Cognito and username/password login.

## [Create authentication service](https://docs.amplify.aws/start/getting-started/auth/q/integration/angular/" \l "create-authentication-service)

amplify add auth

? Do you want to use the default authentication and security configuration? Default configuration

? How do you want users to be able to sign in? Username

? Do you want to configure advanced settings? No, I am done.

copy

To deploy the service, run the push command:

amplify push

copy

Now, the authentication service has been deployed and you can start using it. To view the deployed services in your project at any time, go to Amplify Console by running the following command:

amplify console

copy

## [Create login UI](https://docs.amplify.aws/start/getting-started/auth/q/integration/angular/" \l "create-login-ui)

Now that you have your authentication service deployed to AWS, it's time to add authentication to your Angular app. Creating the login flow can be quite difficult and time consuming to get right. Luckily Amplify has an authentication UI component you can use that will provide the entire authentication flow for us, using your configuration specified in your aws-exports.js file.

First, import Authenticator style inside your stylesheet (e.g. src/style.css):

@import '~@aws-amplify/ui-angular/theme.css';

copy

Navigate to src/app/app.component.html, and wrap app-restaurants with <amplify-authenticator> to enable authentication:

<amplify-authenticator>

<ng-template

amplifySlot="authenticated"

let-user="user"

let-signOut="signOut"

>

<app-restaurants></app-restaurants>

<button (click)="signOut()">Sign Out</button>

</ng-template>

</amplify-authenticator>

copy

### [Unhandled Promise Console Warnings](https://docs.amplify.aws/start/getting-started/auth/q/integration/angular/" \l "unhandled-promise-console-warnings)

If you see Unhandled Promise rejection: [message] ; zone: <root> ; ... errors on console, you can add the following code after you import zone.js inside src/polyfills.ts file:

import 'zone.js/dist/zone'; // Included with Angular CLI.

(window as any).Zone['\_\_zone\_symbol\_\_ignoreConsoleErrorUncaughtError'] = true;

copy

Run the app to see the new Authentication flow protecting the app:

npm start

copy

Now you should see the app load with an authentication flow allowing users to sign up and sign in.

In this example, you used the **Amplify Angular UI** library and the amplify-authenticator component to quickly get up and running with a real-world authentication flow.

You can also [customize](https://ui.docs.amplify.aws/angular/components/authenticator/customization) this component to add or remove fields, update styling, or other configurations.

In addition to the amplify-authenticator you can build custom authentication flows using the Auth class.

Auth has over 30 methods including signUp, signIn, forgotPassword, and signOut that allow you full control over all aspects of the user authentication flow. Check out the complete API [here](https://aws-amplify.github.io/amplify-js/api/classes/authclass.html)

In the next section, you'll host your app on the **Amplify Console**, a hosting service complete with a globally available CDN, atomic deployments, easy custom domains, and CI / CD.

# Deploy and host app

You've successfully built your first app with Amplify! Now that you've built something, it's time to deploy it to the web with Amplify Hosting!

## [Add hosting to your app](https://docs.amplify.aws/start/getting-started/hosting/q/integration/angular/" \l "add-hosting-to-your-app)

You can manually deploy your web app or setup automatic continuous deployment. In this guide we'll cover how to manually deploy and host your static web app to quickly share with others. If you want to learn about continuous deployment instead, please follow [this guide](https://docs.aws.amazon.com/amplify/latest/userguide/multi-environments.html#standard).

From the root of your project, run the following command and select the **bolded options**.

amplify add hosting

copy

? Select the plugin module to execute: # Hosting with Amplify Console (Managed hosting with custom domains, Continuous deployment)

? Choose a type: # Manual Deployment

## [Publish your app](https://docs.amplify.aws/start/getting-started/hosting/q/integration/angular/" \l "publish-your-app)

Run the following command to publish your app.

amplify publish

copy

If you get "Error: bundle initial exceeded maximum budget" error based on bundle size, you need to bump the maximum bundle size budget for your Angular application. Find the `budgets` keyword in `angular.json`, and bump `maximumError`:

{

// ...

"budgets": [

{

"type": "initial",

"maximumWarning": "500kb",

"maximumError": "2mb"

},

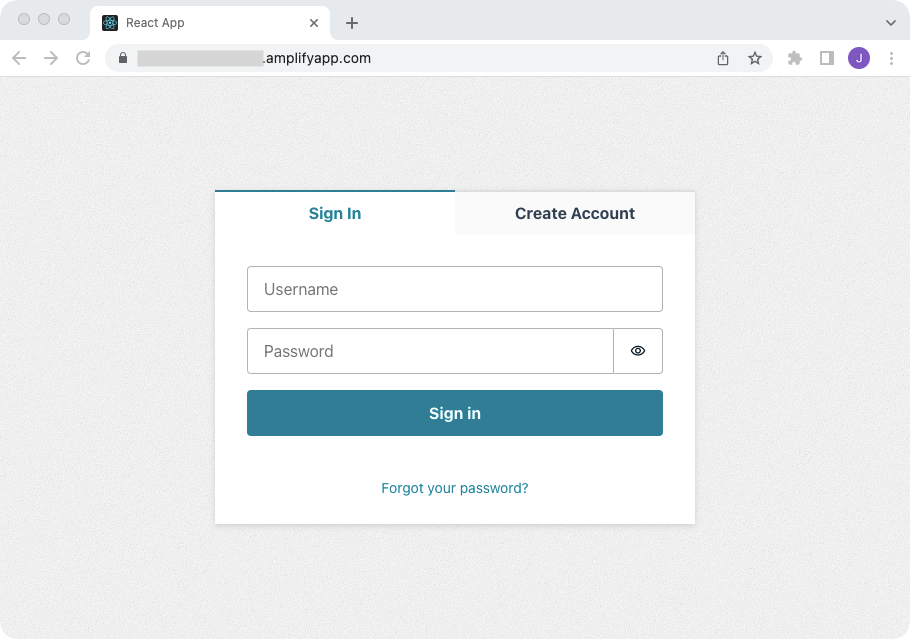
// ...

]

}

copy

Congratulations, your app is online!



After publishing, your terminal will display your app URL hosted on a amplifyapp.com domain. Whenever you have additional changes to publish, just re-run the amplify publish command.

You may get an "AccessDenied" error if your app's distribution directory is not set properly. To fix this, change the distribution directory via amplify configure project and then re-run amplify publish. Distribution directory should match dist/amplify-app.

To view your app and hosting configuration in the Amplify Console, run the amplify console command.

Note: To delete all the environments of the project from the cloud and wipe out all the local files created by Amplify CLI, run amplify delete command.