Cloud-Native Full-Stack Application using AWS Amplify & Next.js

Project Documentation
AWS Amplify + Next.js (App Router + Client Components).
This documentation describes the complete development and deployment process of a full-stack web application using AWS Amplify (Gen 2) and Next.js App Router (Client Components).

---

1) Project Overview

This project is a full-stack, cloud-backed web application built with Next.js (App Router + Client Components), powered by AWS Amplify for backend services and hosting.
It demonstrates how to combine modern frontend frameworks with cloud-managed backend services — including APIs, database, authentication, and hosting — using Amplify.

---

2) Objective

The main purpose of this documentation is to explain:
✓ How to initialize a cloud backend using AWS Amplify
✓ How to configure a Next.js application to interact with that backend
✓ How to build and deploy the app to the cloud using Amplify Hosting
✓ Why each step is necessary in a cloud-native project

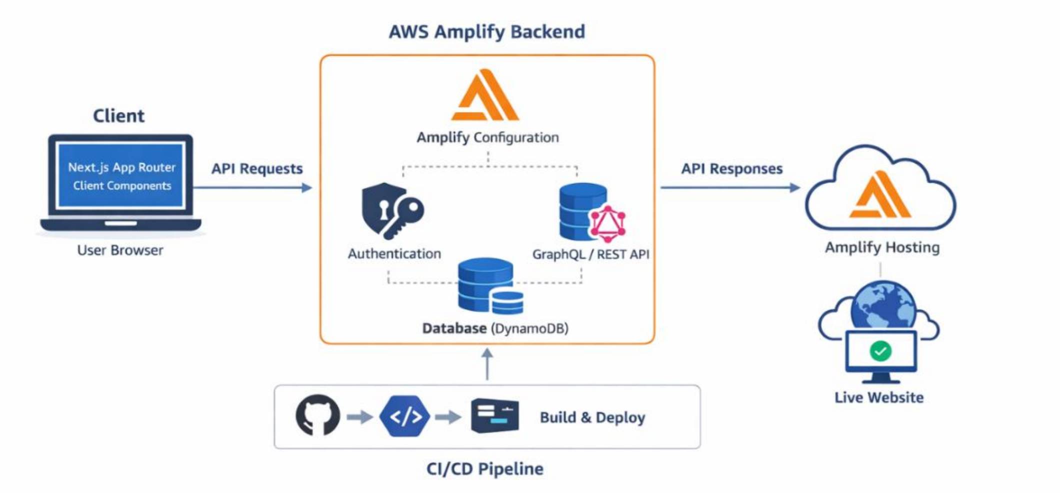
---

3) Tech Stack

Component	Technology
Frontend UI	Next.js (App Router + Client Components)
Backend	AWS Amplify (Gen 2)
API	AWS AppSync / REST (via Amplify)
Hosting	Amplify Hosting
Language	TypeScript / JavaScript
Deployment Automation	CI/CD Workflow (Amplify)

---

4) Project Architecture



Explanation:

- Frontend (Next.js): Renders UI; uses client components to interact with backend APIs.
- Amplify Backend: Manages serverless resources (API, database, authentication).
- CI/CD Pipeline: Automatically builds and deploys the app when code is pushed to the repository.
- Hosting: Amplify Hosting serves the web app globally with edge caching.

---

5) Pre-Requisites

Before starting the project, ensure the following tools are installed:

- Node.js v14.x or later
- npm v6.14.4 or later
- Git v2.14.1 or later

Why these are required

- Node.js & npm: Required to run and manage the Next.js application and dependencies.
- Git: Enables version control and integration with Amplify's CI/CD pipeline.
- Basic React & Next.js knowledge: Helps understand App Router, components, and data flow.

---

a. Frontend Configuration with Amplify

The Amplify configuration file (aws-exports.js or amplify_outputs.json) is imported into the Next.js application. This allows the frontend to automatically discover backend endpoints, authentication flows, and region-specific settings.
Amplify is configured globally so all components can access backend services.

Why this step is required

- Makes backend accessible to frontend
- Centralizes cloud configuration
- Prevents hardcoding credentials

b. API & Database Creation

AWS Amplify is used to create a GraphQL API backed by Amazon DynamoDB. Amplify automatically generates schemas, resolvers, and permissions without manual server provisioning.
This follows a serverless architecture, where AWS manages scaling and availability.

Why this step is required

- Enables data storage and retrieval
- Removes need for manual server setup
- Provides scalable backend

c. Hosting & Initial Deployment

The application repository is connected to AWS Amplify Hosting. Amplify automatically detects the framework and sets up a CI/CD pipeline that builds and deploys the application whenever code is pushed.
The deployed application is hosted globally using a CDN.

Why this step is required

- Automates deployment
- Ensures high availability
- Reduces manual errors.

d. Local Development Environment Setup

The deployed backend configuration is downloaded as amplify_outputs.json and connected to the local project. This allows developers to work locally while using real cloud resources.
Local testing ensures changes work correctly before pushing to production.

Why this step is required

- Enables safe development
- Maintains consistency between environments
- Improves debugging Cloud

e. Implementing Delete Functionality

Frontend logic is extended to add delete functionality for to-do items. The delete operation interacts with the backend API and updates the database accordingly.
This demonstrates how frontend components trigger backend operations.

Why this step is required

- Enhances application functionality
- Demonstrates API integration
- Validates frontend–backend sync

f. Authentication Implementation

Authentication is implemented using Amazon Cognito via AWS Amplify Auth. The Authenticator UI component automatically handles login, signup, email verification, and session management.
This ensures secure access control without custom authentication logic.

Why this step is required

- Secures application access
- Manages users centrally
- Supports scalable authentication

g. CI/CD Pipeline

CI/CD starts **when code is pushed to GitHub**. AWS Amplify automatically:

1. Pulls latest code
2. Installs dependencies
3. Builds the application
4. Deploys backend and frontend
5. Publishes to hosting environment

This ensures continuous integration and continuous deployment.

Why this step is required

- Faster deployments
- Reduced human errors
- Industry-standard DevOps practice

h. IAM & Backend Permissions

IAM credentials are configured locally to allow backend updates. The IAM user is granted limited permissions using **AmplifyBackendDeployFullAccess**, ensuring secure backend deployments.
This follows the principle of least privilege.

Why this step is required

- Secure backend updates
- Controlled access to AWS resources
- Prevents unauthorized changes

i. Cloud Sandbox Environment

A cloud sandbox is an **isolated backend environment** used for development and testing. It allows backend changes without affecting production users.

In this project, sandbox is used to safely implement authorization rules.
Why this step is required

- Protects production backend
- Enables safe experimentation
- Supports multi-user development

j. Per-User Authorization (Owner-Based Rules)

Owner-based authorization ensures that **each user can access only their own data**. Authorization rules are enforced at the API level using authentication tokens.
This is a critical security feature in real-world applications.
Why this step is required

- Prevents data leakage
- Enforces access control
- Aligns with enterprise security standards

k. Final Deployment & Verification

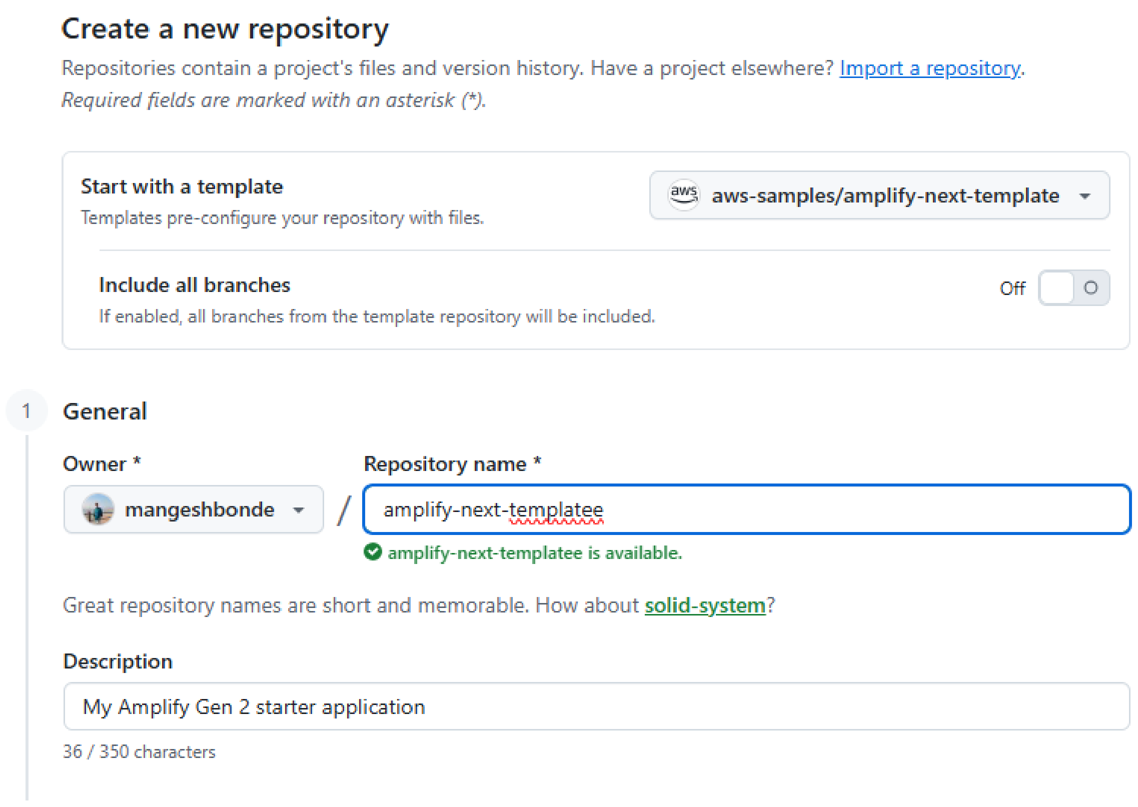
After testing in sandbox, changes are pushed to the main branch. Amplify CI/CD redeploys the application, and functionality is verified in the live environment.
This completes the **end-to-end cloud deployment lifecycle**.
Why this step is required

- Ensures production readiness
- Validates all integrations
- Confirms system stability

---

1. Create the repository

Use our starter template to create a repository in your GitHub account. This template scaffolds create-next-app with Amplify backend capabilities. These template contain the code of todo app from aws samples.
Use the form in GitHub to finalize your repo's creation.



Create a new repository
Repositories contain a project's files and version history. Have a project elsewhere? Import a repository.
Required fields are marked with an asterisk (*).

Start with a template
Templates pre-configure your repository with files.
aws-samples/amplify-next-template

Include all branches
If enabled, all branches from the template repository will be included.
Off

1 General

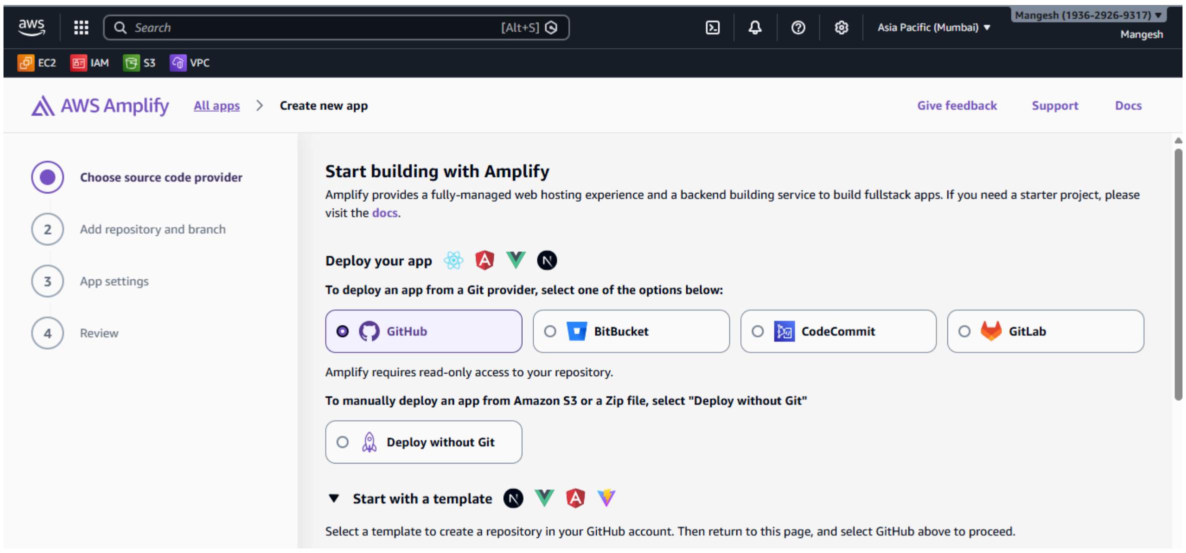
Owner *
mangeshbonde
/
Repository name *
amplify-next-templatee
amplify-next-templatee is available.

Great repository names are short and memorable. How about solid-system?

Description
My Amplify Gen 2 starter application
36 / 350 characters

2. Deploy the starter app

Now that the repository has been created, deploy it with Amplify.
Select **Start with an existing app** > **GitHub**. After you give Amplify access to your GitHub account via the popup window, pick the repository and main branch to deploy. Make no other changes and click through the flow to **Save and deploy**.



Start building with Amplify

Amplify provides a fully-managed web hosting experience and a backend building service to build fullstack apps. If you need a starter project, please visit the docs.

Deploy your app

To deploy an app from a Git provider, select one of the options below:

[✓ GitHub] BitBucket CodeCommit GitLab

Amplify requires read-only access to your repository.

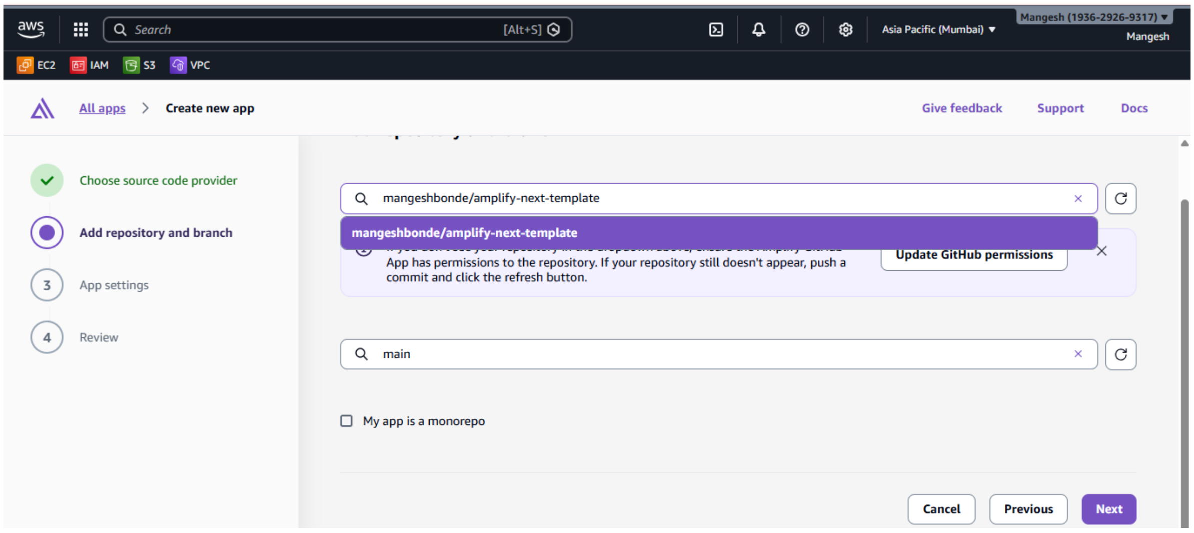
To manually deploy an app from Amazon S3 or a Zip file, select "Deploy without Git"

Deploy without Git

Start with a template

Select a template to create a repository in your GitHub account. Then return to this page, and select GitHub above to proceed.

Then next..



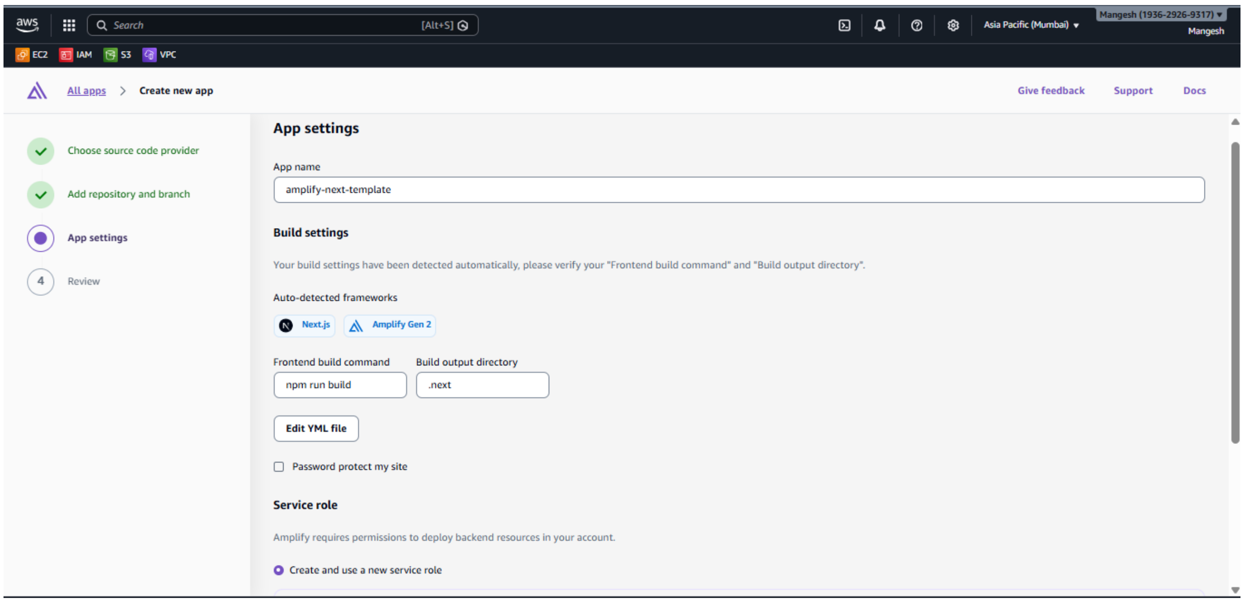
mangeshbonde/amplify-next-template

App has permissions to the repository. If your repository still doesn't appear, push a commit and click the refresh button.

main

My app is a monorepo

Then next.. You will see the Overview



App settings

App name: amplify-next-template

Build settings

Your build settings have been detected automatically, please verify your "Frontend build command" and "Build output directory".

Auto-detected frameworks: Next.js, Amplify Gen 2

Frontend build command: npm run build

Build output directory: .next

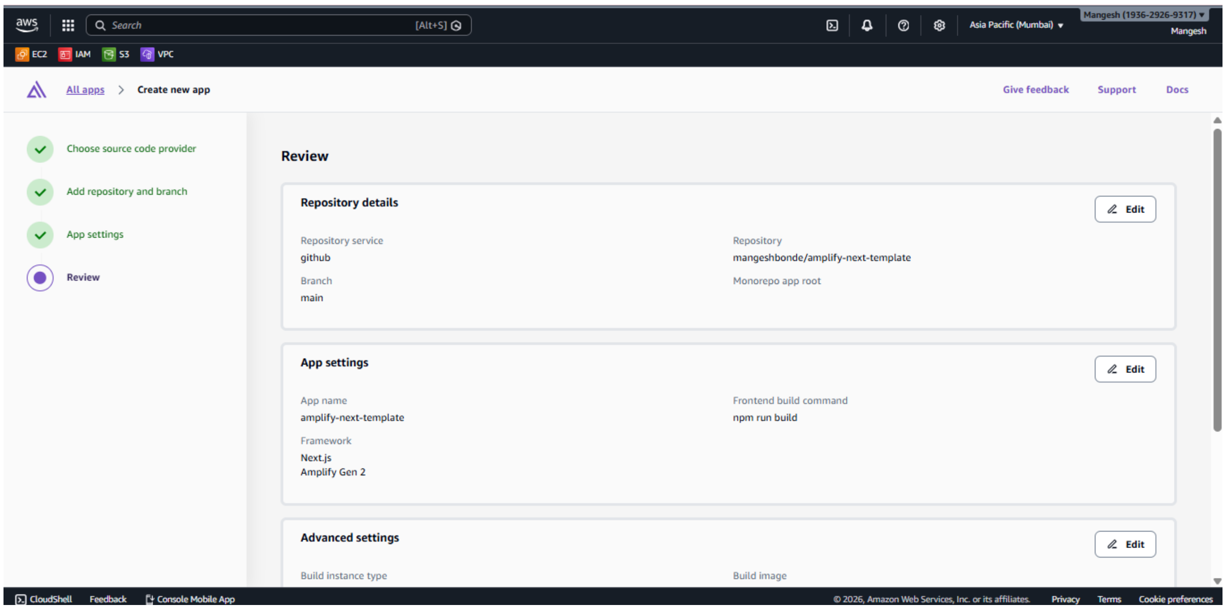
Password protect my site

Service role

Amplify requires permissions to deploy backend resources in your account.

[✓ Create and use a new service role]

Then click Save and Deploy..



Review

Repository details

Repository service
github

Repository
mangeshbonde/amplify-next-template

Branch
main

Monorepo app root

App settings

App name
amplify-next-template

Frontend build command
npm run build

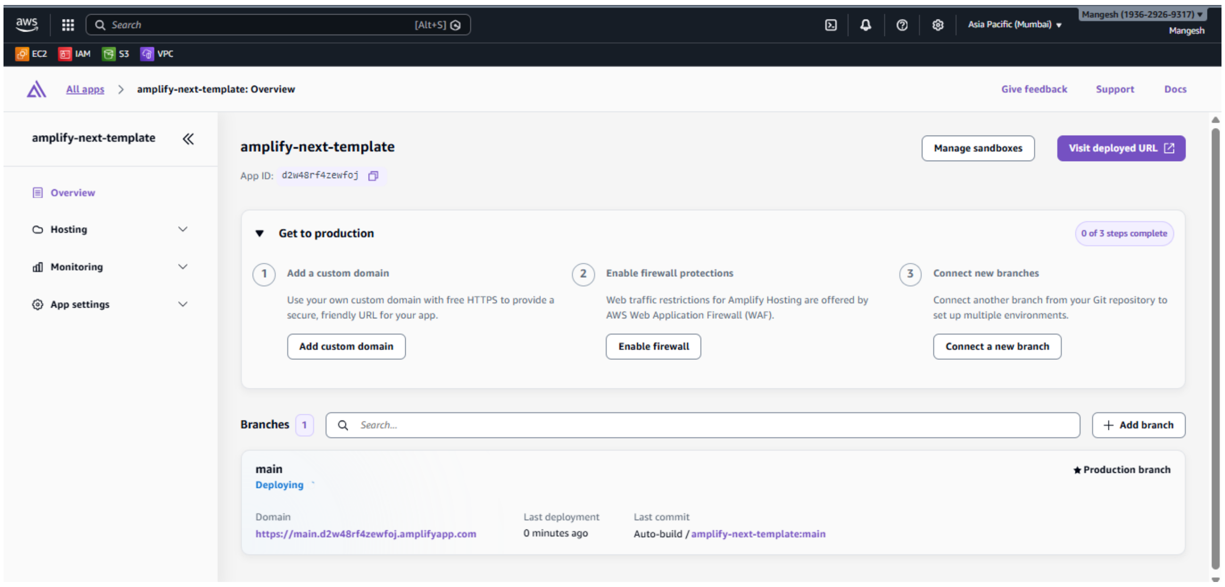
Framework
Next.js
Amplify Gen 2

Advanced settings

Build instance type

Build image

We can see here process is Deploying…



amplify-next-template

App ID: d2w48rf4zewfoj

Get to production — 0 of 3 steps complete

1 Add a custom domain
Use your own custom domain with free HTTPS to provide a secure, friendly URL for your app.

2 Enable firewall protections
Web traffic restrictions for Amplify Hosting are offered by AWS Web Application Firewall (WAF).

3 Connect new branches
Connect another branch from your Git repository to set up multiple environments.

Branches 1

main
Deploying
★ Production branch

Domain
https://main.d2w48rf4zewfoj.amplifyapp.com

Last deployment
0 minutes ago

Last commit
Auto-build / amplify-next-template:main

CI/CD Pipeline with Amplify Hosting

Connect to Git

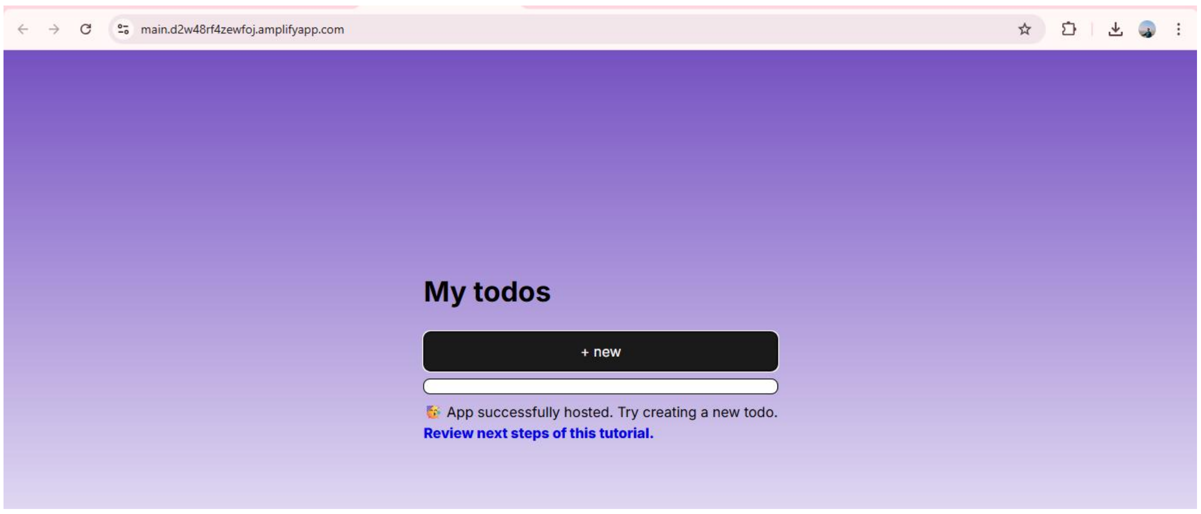
1. Push your local repo to GitHub / GitLab.
2. Open Amplify Console → Deploy App.
3. Connect your repo and branch.

Amplify Hosting automatically builds and deploys your app every time you push changes. This is the core of CI/CD.

3. View deployed app

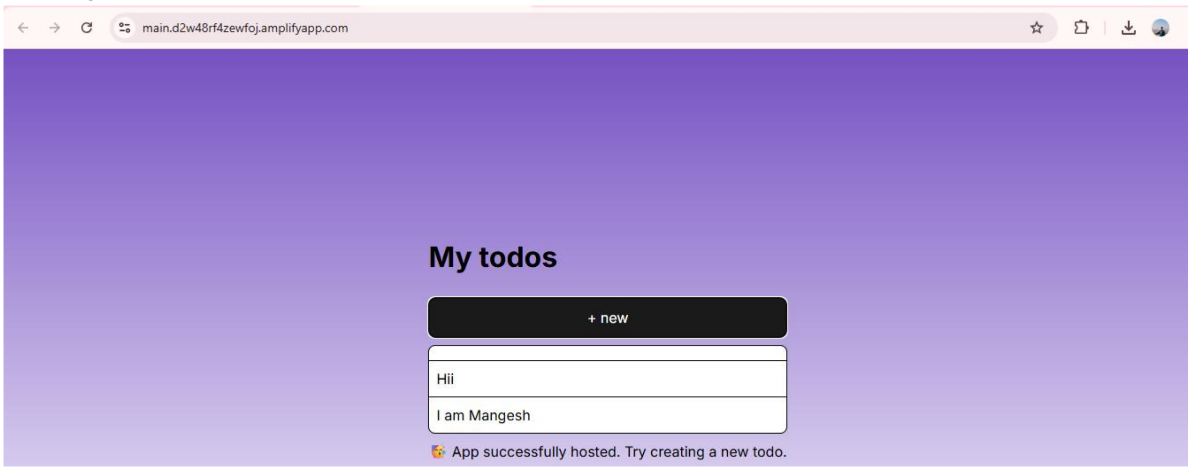
When the build completes, visit the newly deployed branch by selecting "View deployed URL". Since the build deployed an API, database, and authentication backend, you will be able to create new to-do items.

https://main.d2w48rf4zewfoj.amplifyapp.com/



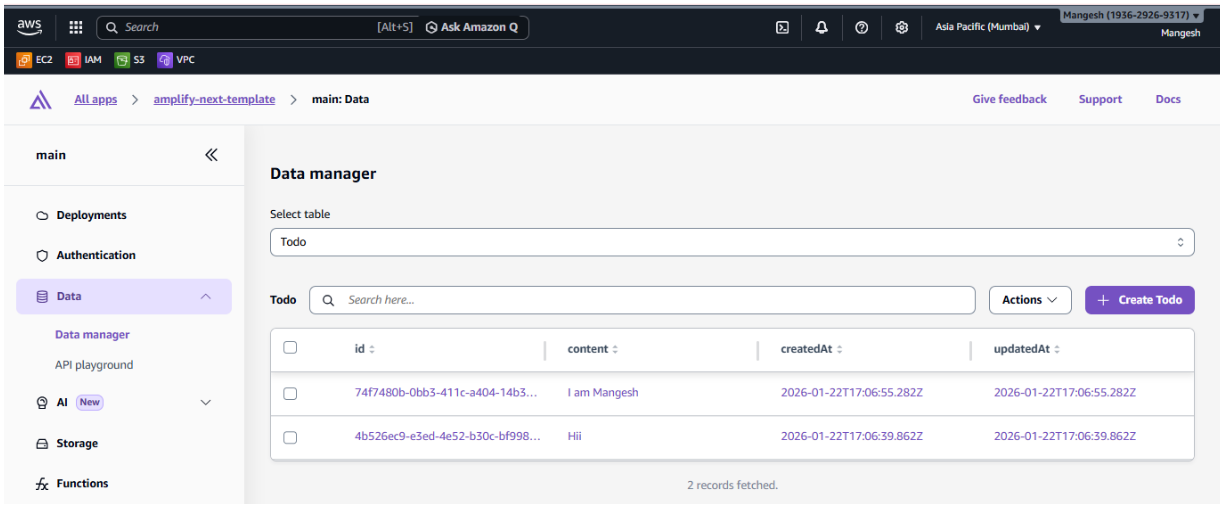
🥳 App successfully hosted. Try creating a new todo.
Review next steps of this tutorial.

Adding todo list



🥳 App successfully hosted. Try creating a new todo.

In the Amplify console, click into the deployment branch (in this case **main**) > select **Data** in the left-hand menu > **Data manager** to see the data entered in your database.



Todo

	id	content	createdAt	updatedAt
☐	74f7480b-0bb3-411c-a404-14b3...	I am Mangesh	2026-01-22T17:06:55.282Z	2026-01-22T17:06:55.282Z
☐	4b526ec9-e3ed-4e52-b30c-bf998...	Hii	2026-01-22T17:06:39.862Z	2026-01-22T17:06:39.862Z

2 records fetched.

Make frontend updates

Let's learn how to enhance the app functionality by creating a delete flow for to-do list items.

Frontend Configuration with Amplify

The Amplify configuration file (aws-exports.js or amplify_outputs.json) is imported into the Next.js application. This allows the frontend to automatically discover backend endpoints, authentication flows, and region-specific settings.

Amplify is configured globally so all components can access backend services.

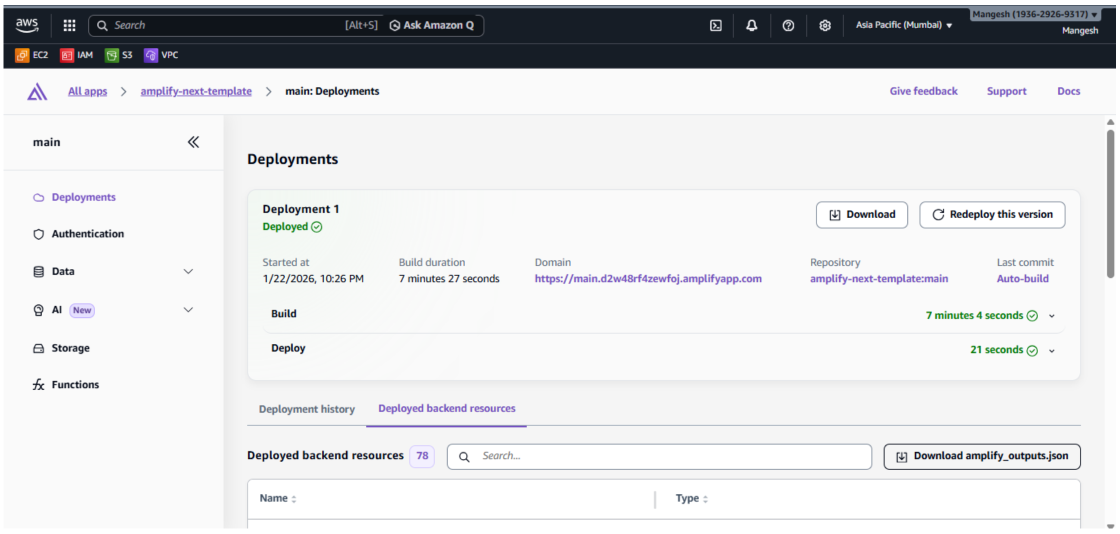
Why this step is required

- Makes backend accessible to frontend

- Centralizes cloud configuration
- Prevents hardcoding credentials

Set up local environment

Now let's set up our local development environment to add features to the frontend. Click on your deployed branch and you will land on the **Deployments** page which shows you your build history and a list of deployed backend resources.



All apps > amplify-next-template > main: Deployments

Deployments

Deployment 1
Deployed

Started at: 1/22/2026, 10:26 PM
Build duration: 7 minutes 27 seconds
Domain: https://main.d2w48rf4zewfoj.amplifyapp.com
Repository: amplify-next-template:main
Last commit: Auto-build

Build — 7 minutes 4 seconds
Deploy — 21 seconds

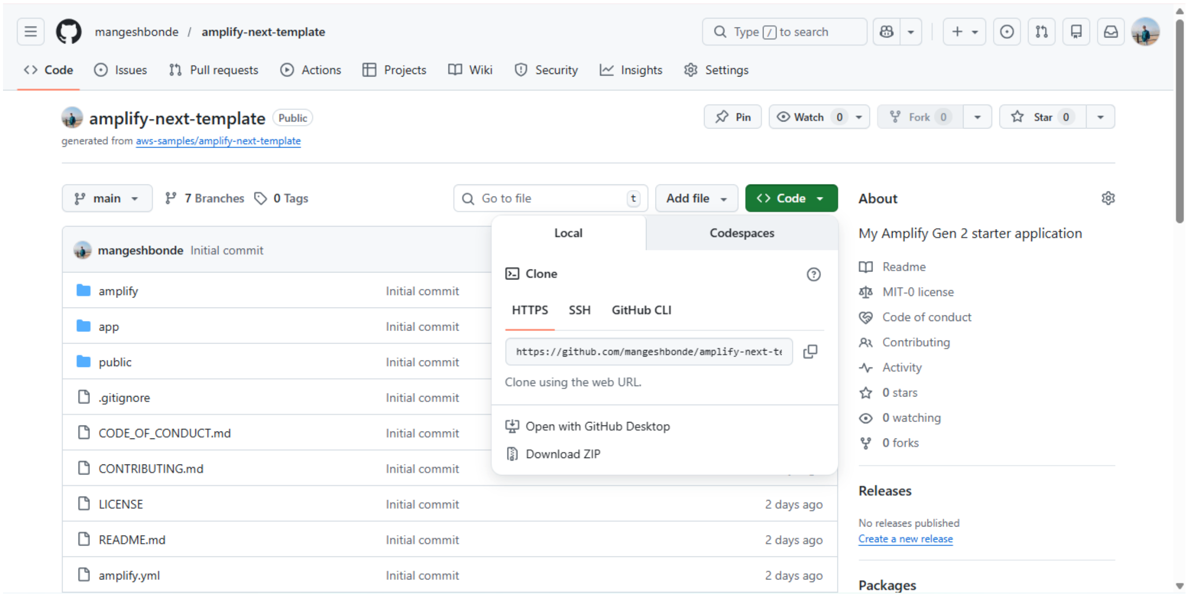
Deployment history | Deployed backend resources

Deployed backend resources 78 | Download amplify_outputs.json

Name	Type

At the bottom of the page you will see a tab for **Deployed backend resources**. Click on the tab and then click the **Download amplify_outputs.json file** button.

Clone the repository locally in VS code.
Here copy the link



mangeshbonde / amplify-next-template

amplify-next-template Public
generated from aws-samples/amplify-next-template

main | 7 Branches | 0 Tags

mangeshbonde Initial commit

amplify	Initial commit
app	Initial commit
public	Initial commit
.gitignore	Initial commit
CODE_OF_CONDUCT.md	Initial commit
CONTRIBUTING.md	Initial commit
LICENSE	Initial commit	2 days ago
README.md	Initial commit	2 days ago
amplify.yml	Initial commit	2 days ago

Clone — HTTPS
https://github.com/mangeshbonde/amplify-next-te
Clone using the web URL.
Open with GitHub Desktop
Download ZIP

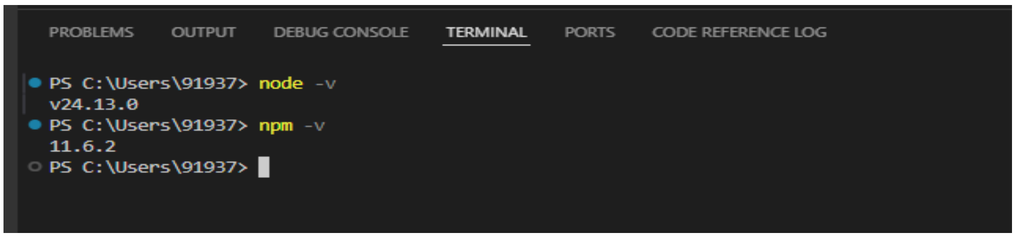
About
My Amplify Gen 2 starter application
Readme
MIT-0 license
Code of conduct
Contributing
Activity
0 stars
0 watching
0 forks

Releases
No releases published
Create a new release

Packages

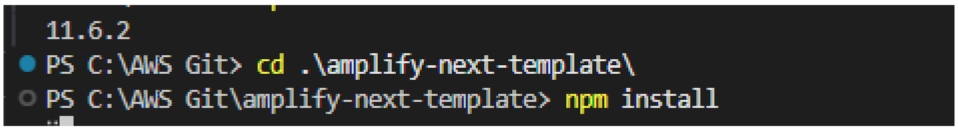
Go into VS code Terminal
git clone https://github.com/mangeshbonde/amplify-next-template.git   #Command

Make sure you have installed node.js and node package manager (npm)



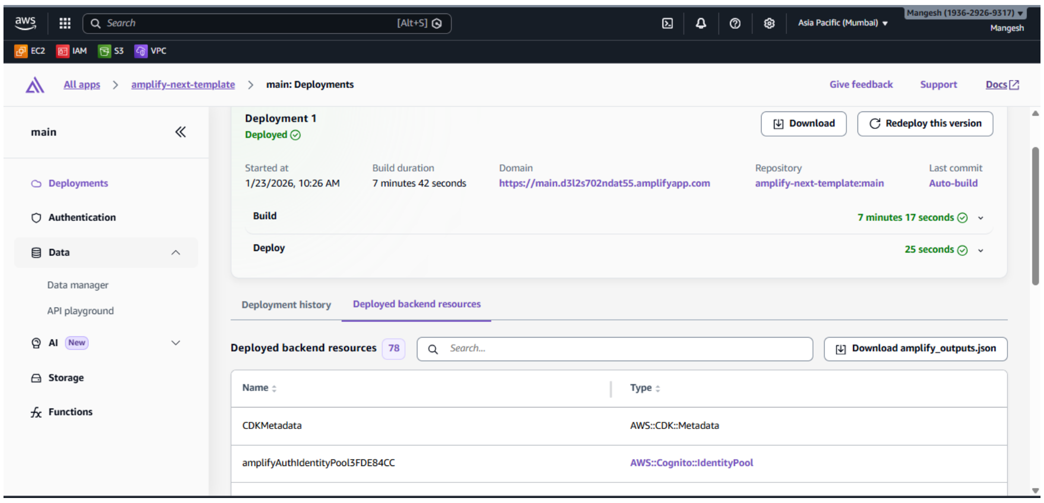
```
PS C:\Users\91937> node -v
v24.13.0
PS C:\Users\91937> npm -v
11.6.2
PS C:\Users\91937>
```

cd amplify-next-template && npm install



```
11.6.2
PS C:\AWS Git> cd .\amplify-next-template\
PS C:\AWS Git\amplify-next-template> npm install
```

Download the file from your deployed amplify app → main →Deployed Backend Resource →Download amplify_outputs.json



Deployment 1
Deployed

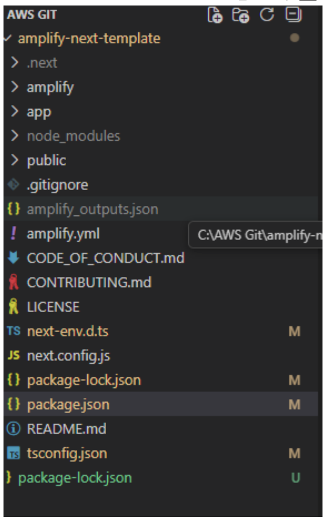
Started at: 1/23/2026, 10:26 AM
Build duration: 7 minutes 42 seconds
Domain: https://main.d3l2s702ndat55.amplifyapp.com
Repository: amplify-next-template:main
Last commit: Auto-build

Build 7 minutes 17 seconds
Deploy 25 seconds

Deployed backend resources 78

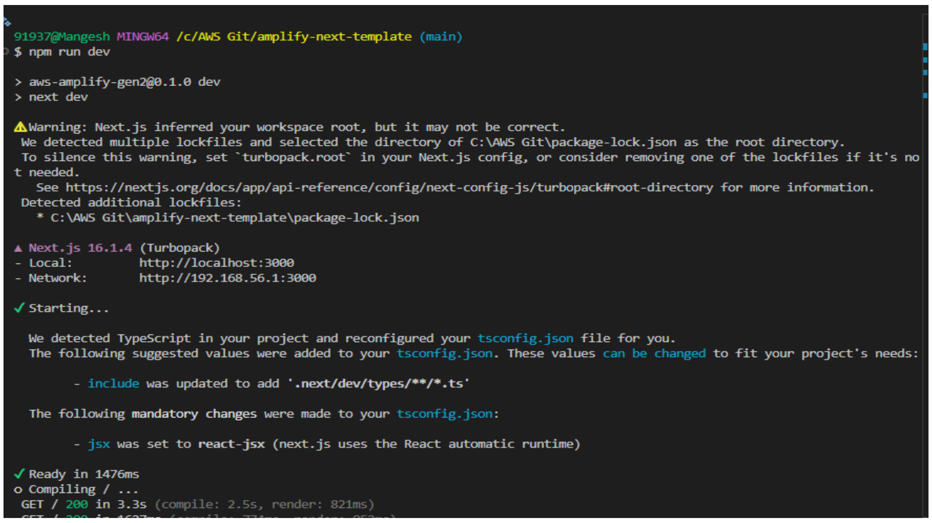
Name	Type
CDKMetadata	AWS::CDK::Metadata
amplifyAuthIdentityPool3FDE84CC	AWS::Cognito::IdentityPool

Now move the amplify_outputs.json file you downloaded above to the root of your project.



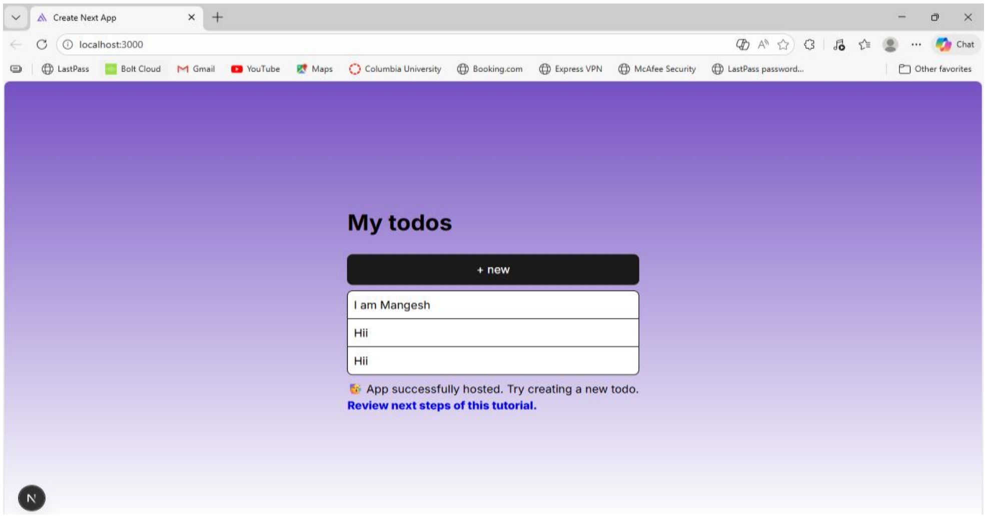
```
AWS GIT
amplify-next-template
  > .next
  > amplify
  > app
  > node_modules
  > public
  .gitignore
  amplify_outputs.json
  amplify.yml                C:\AWS Git\amplify-n
  CODE_OF_CONDUCT.md
  CONTRIBUTING.md
  LICENSE
  next-env.d.ts              M
  next.config.js
  package-lock.json          M
  package.json               M
  README.md
  tsconfig.json              M
package-lock.json            U
```

The *amplify_outputs.json* file contains backend endpoint information, publicly-viewable API keys, authentication flow information, and more. The Amplify client library uses this outputs file to connect to your Amplify Backend. You can review how the outputs file is imported within the main.tsx file and then passed into the Amplify.configure(...) function of the Amplify client library.
Installing required libraries for the project to run.
Then add command to run project *npm run dev*



```
91937@Mangesh MINGW64 /c/AWS Git/amplify-next-template (main)
$ npm run dev

> aws-amplify-gen2@0.1.0 dev
> next dev

⚠Warning: Next.js inferred your workspace root, but it may not be correct.
 We detected multiple lockfiles and selected the directory of C:\AWS Git\package-lock.json as the root directory.
 To silence this warning, set `turbopack.root` in your Next.js config, or consider removing one of the lockfiles if it's no
t needed.
   See https://nextjs.org/docs/app/api-reference/config/next-config-js/turbopack#root-directory for more information.
 Detected additional lockfiles:
   * C:\AWS Git\amplify-next-template\package-lock.json

▲ Next.js 16.1.4 (Turbopack)
- Local:        http://localhost:3000
- Network:      http://192.168.56.1:3000

✓ Starting...

  We detected TypeScript in your project and reconfigured your tsconfig.json file for you.
  The following suggested values were added to your tsconfig.json. These values can be changed to fit your project's needs:

        - include was updated to add '.next/dev/types/**/*.ts'

  The following mandatory changes were made to your tsconfig.json:

        - jsx was set to react-jsx (next.js uses the React automatic runtime)

✓ Ready in 1476ms
o Compiling / ...
 GET / 200 in 3.3s (compile: 2.5s, render: 821ms)
```

Click the link Local → We see todo app which we have cloned in our local computer.



My todos

+ new

I am Mangesh

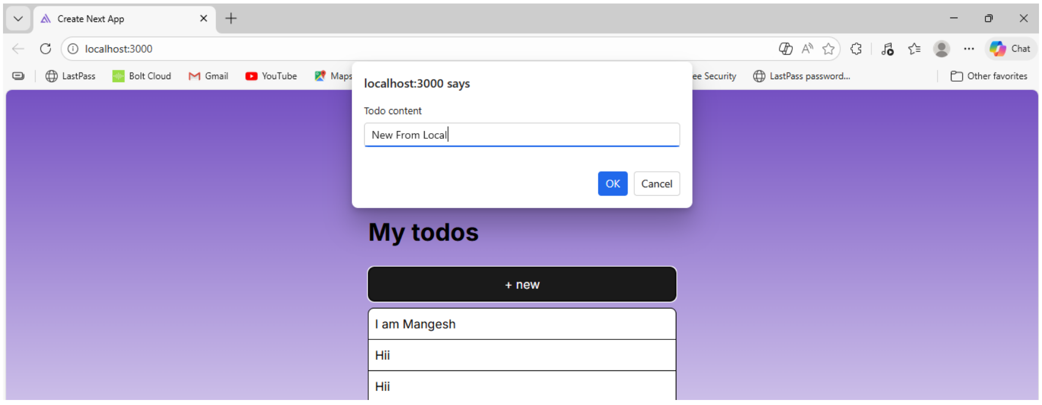
Hii

Hii

App successfully hosted. Try creating a new todo.

Review next steps of this tutorial.

Add Todo from local environment.



localhost:3000 says

Todo content

New From Local

OK Cancel

My todos

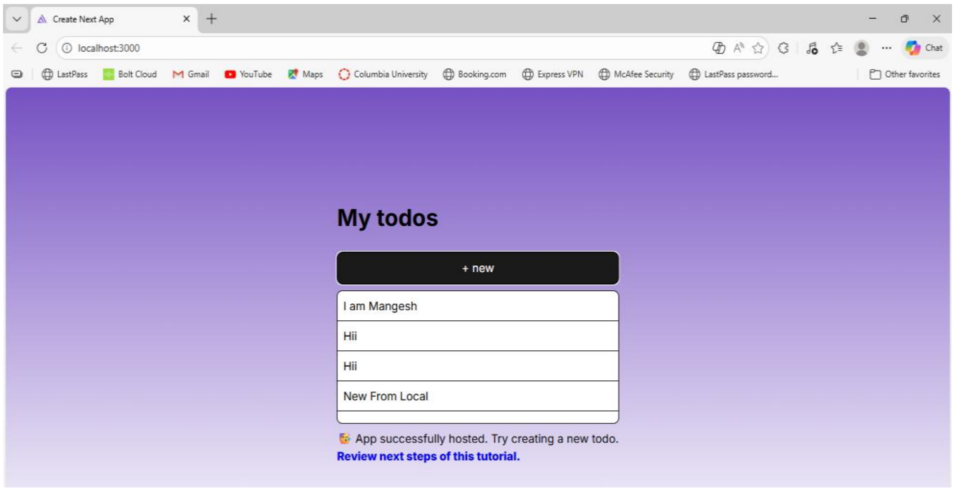
+ new

I am Mangesh

Hii

Hii

Added..



My todos

+ new

I am Mangesh

Hii

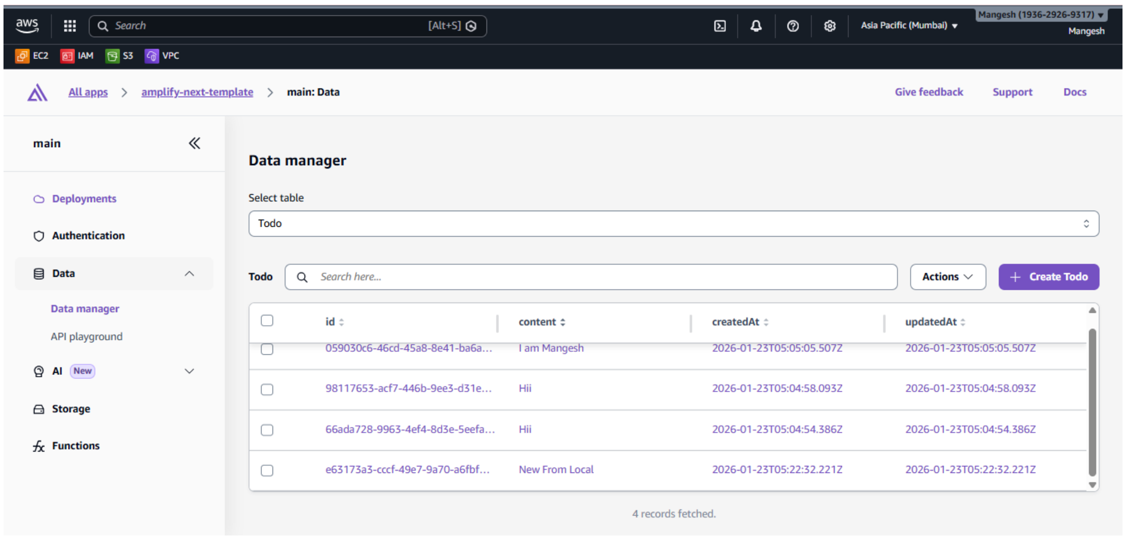
Hii

New From Local

App successfully hosted. Try creating a new todo.

Review next steps of this tutorial.

Lets check wheater it reflects in our Amplify service or not ..
It shows . Means our local environment is connected to AWS resource.



Data manager

Select table: Todo

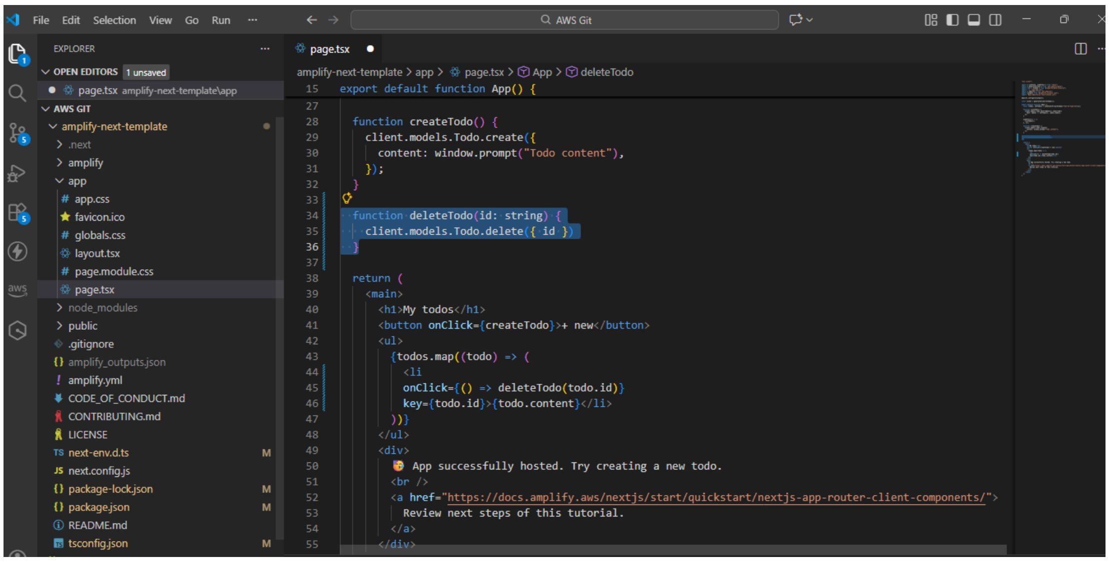
Todo

id	content	createdAt	updatedAt
059030c6-46cd-45a8-8e41-ba6a...	I am Mangesh	2026-01-23T05:05:05.507Z	2026-01-23T05:05:05.507Z
98117653-acf7-446b-9ee3-d31e...	Hii	2026-01-23T05:04:58.093Z	2026-01-23T05:04:58.093Z
66ada728-9963-4ef4-8d3e-5eefa...	Hii	2026-01-23T05:04:54.386Z	2026-01-23T05:04:54.386Z
e63173a3-cccf-49e7-9a70-a6fbf...	New From Local	2026-01-23T05:22:32.221Z	2026-01-23T05:22:32.221Z

4 records fetched.

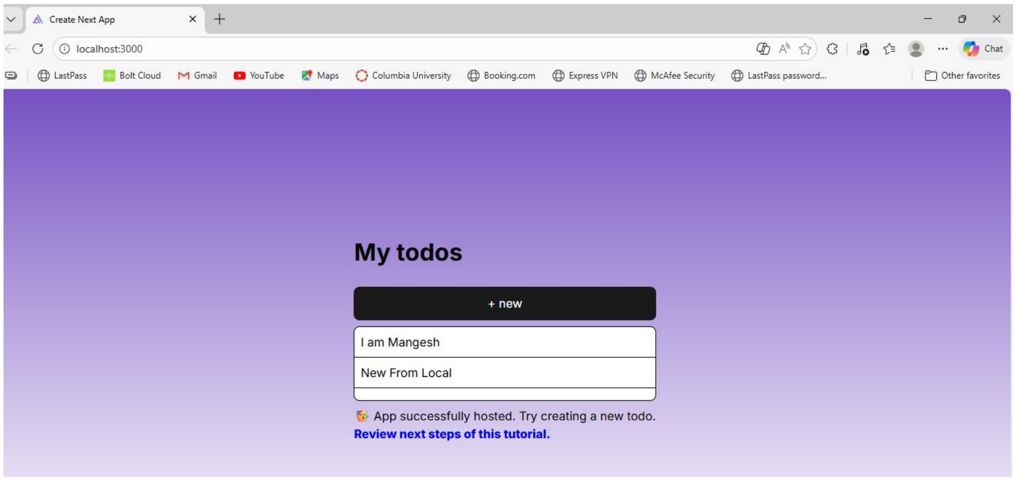
<u>Implement delete functionality</u>

Go to the **app/page.tsx** file and add in a new deleteTodo functionality and pass function into the <li> element's onClick handler.



```
export default function App() {

  function createTodo() {
    client.models.Todo.create({
      content: window.prompt("Todo content"),
    });
  }

  function deleteTodo(id: string) {
    client.models.Todo.delete({ id })
  }

  return (
    <main>
      <h1>My todos</h1>
      <button onClick={createTodo}>+ new</button>
      <ul>
        {todos.map((todo) => (
          <li
          onClick={() => deleteTodo(todo.id)}
          key={todo.id}>{todo.content}</li>
        ))}
      </ul>
      <div>
        🥳 App successfully hosted. Try creating a new todo.
        <br />
        <a href="https://docs.amplify.aws/nextjs/start/quickstart/nextjs-app-router-client-components/">
          Review next steps of this tutorial.
        </a>
      </div>
```

Save the changes and refresh the page .
After click the todo list , you can se it may delete after clicking specific todo list.
It means Delete Function is working.



My todos

+ new

I am Mangesh

New From Local

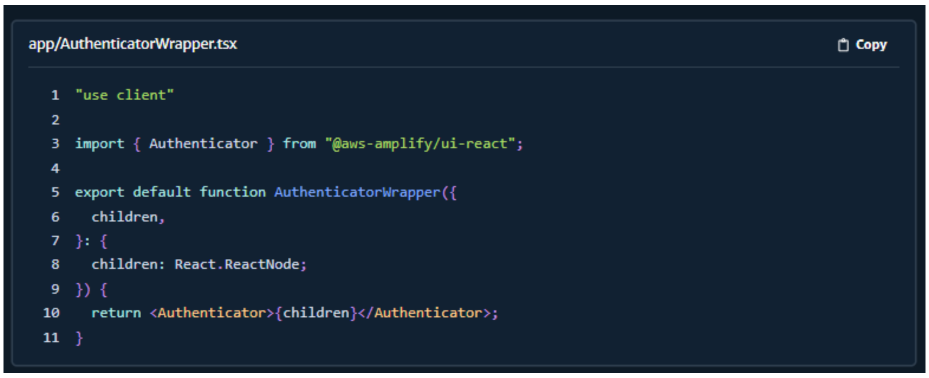
🥳 App successfully hosted. Try creating a new todo.
Review next steps of this tutorial.

Implement login UI

The starter application already has a pre-configured auth backend defined in the **amplify/auth/resource.ts** file. We've configured it to support email and password login but you can extend it to support a variety of login mechanisms, including Google, Amazon, Sign In With Apple, and Facebook.

The fastest way to get your login experience up and running is to use our Authenticator UI component. To properly integrate it with Next.js App Router, we'll create a client component wrapper and use it in the layout.

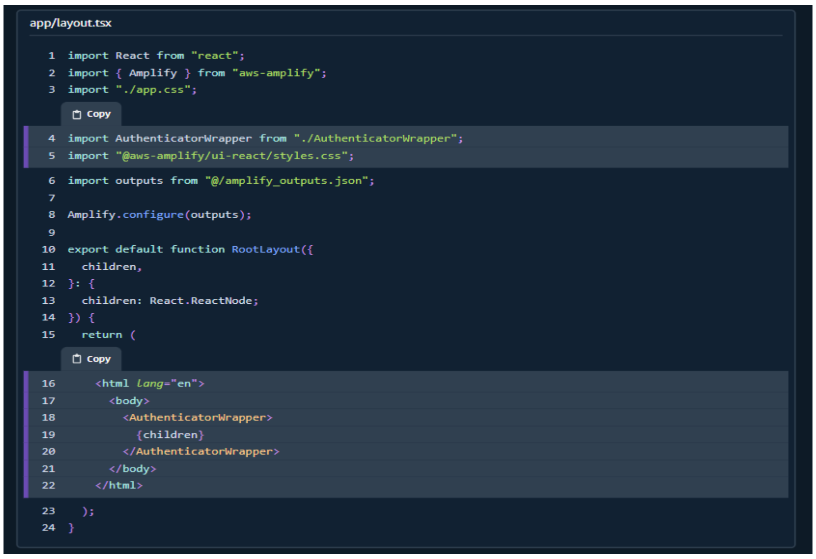
First, create an AuthenticatorWrapper.tsx file in your app directory:



app/AuthenticatorWrapper.tsx

```
"use client"

import { Authenticator } from "@aws-amplify/ui-react";

export default function AuthenticatorWrapper({
  children,
}: {
  children: React.ReactNode;
}) {
  return <Authenticator>{children}</Authenticator>;
}
```

Next, update your app/layout.tsx file to import and use the AuthenticatorWrapper component:

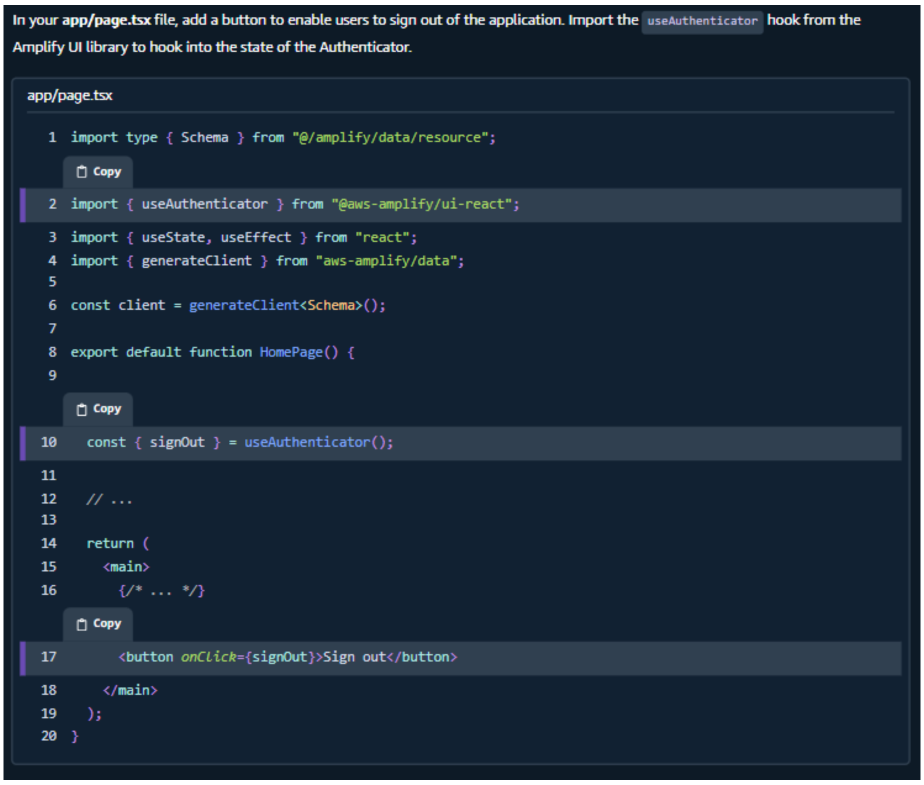


app/layout.tsx

```
import React from "react";
import { Amplify } from "aws-amplify";
import "./app.css";
import AuthenticatorWrapper from "./AuthenticatorWrapper";
import "@aws-amplify/ui-react/styles.css";
import outputs from "@/amplify_outputs.json";

Amplify.configure(outputs);

export default function RootLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  return (
    <html lang="en">
      <body>
        <AuthenticatorWrapper>
          {children}
        </AuthenticatorWrapper>
      </body>
    </html>
  );
}
```

The Authenticator component auto-detects your auth backend settings and renders the correct UI state based on the auth backend's authentication flow.

In your **app/page.tsx** file, add a button to enable users to sign out of the application. Import the [useAuthenticator] hook from the Amplify UI library to hook into the state of the Authenticator.

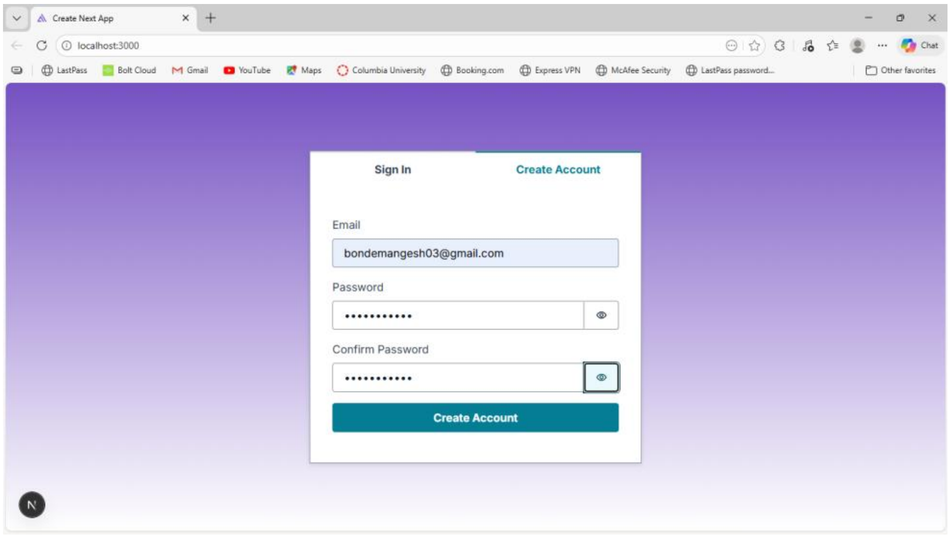

In your **app/page.tsx** file, add a button to enable users to sign out of the application. Import the `useAuthenticator` hook from the Amplify UI library to hook into the state of the Authenticator.

app/page.tsx

```tsx
import type { Schema } from "@/amplify/data/resource";
import { useAuthenticator } from "@aws-amplify/ui-react";
import { useState, useEffect } from "react";
import { generateClient } from "aws-amplify/data";

const client = generateClient<Schema>();

export default function HomePage() {

  const { signOut } = useAuthenticator();

  // ...

  return (
    <main>
      {/* ... */}
      <button onClick={signOut}>Sign out</button>
    </main>
  );
}
```

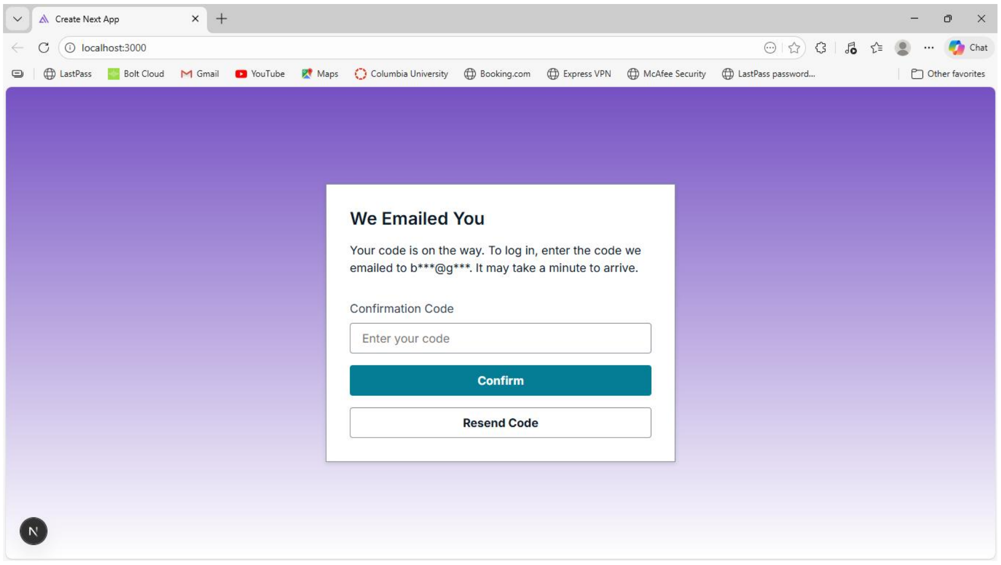
Try out your application in your localhost environment again. You should be presented with a login experience now.

You must create account first



Email
bondemangesh03@gmail.com
Password
•••••••••••
Confirm Password
•••••••••••

For verification it sends OTP to email



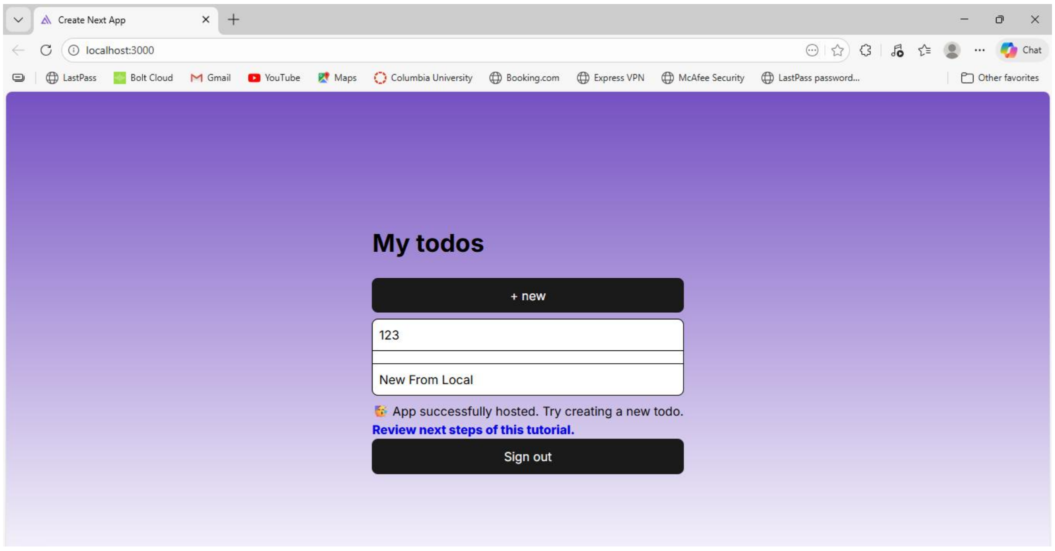
We Emailed You

Your code is on the way. To log in, enter the code we emailed to b***@g***. It may take a minute to arrive.

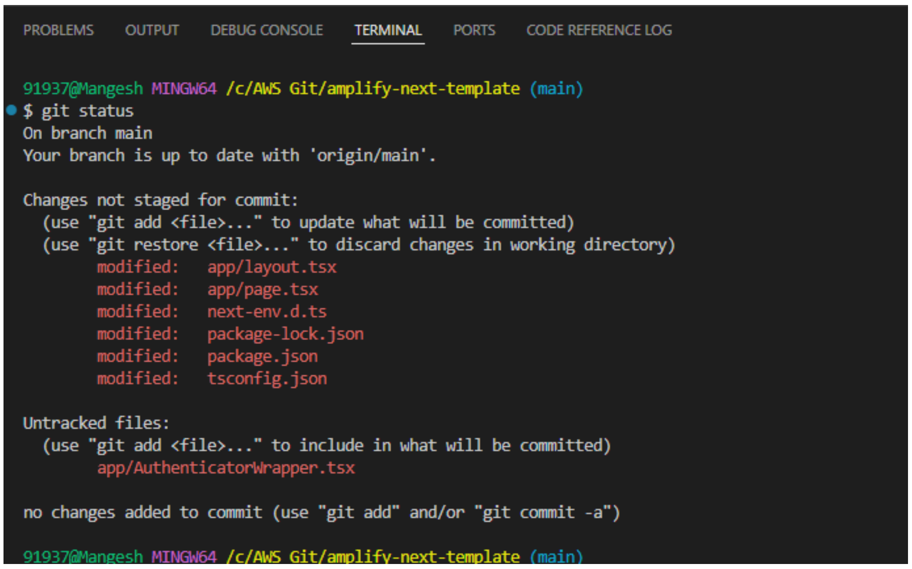
Confirmation Code
Enter your code

Sign in .

We have added sign in and sign out functionality in it.

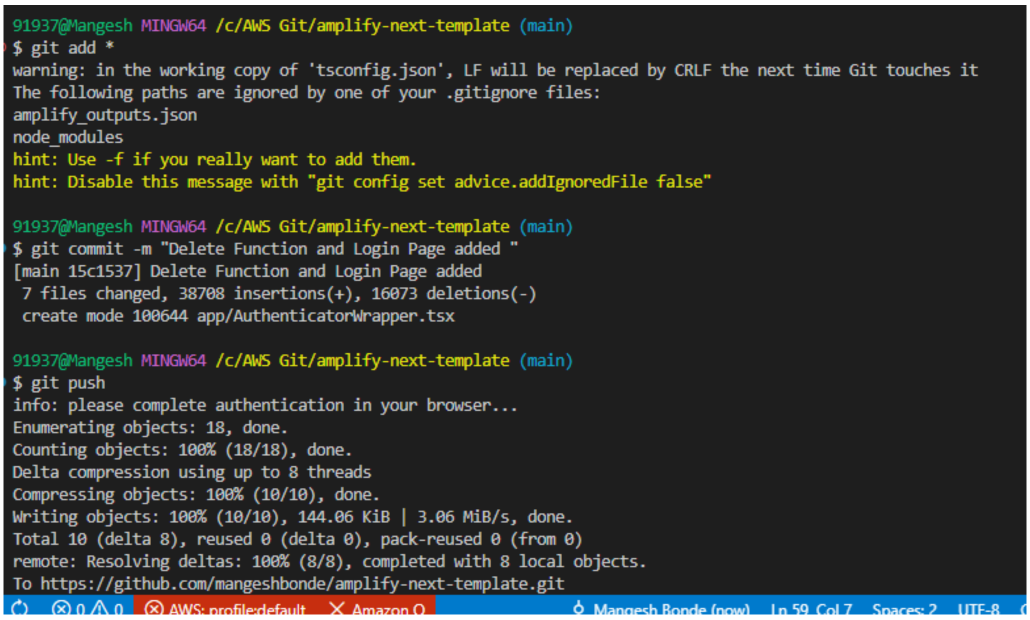

Create Next App
localhost:3000
My todos
+ new
123
New From Local
App successfully hosted. Try creating a new todo.
Review next steps of this tutorial.
Sign out

Check git status in VS code terminal.


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS CODE REFERENCE LOG

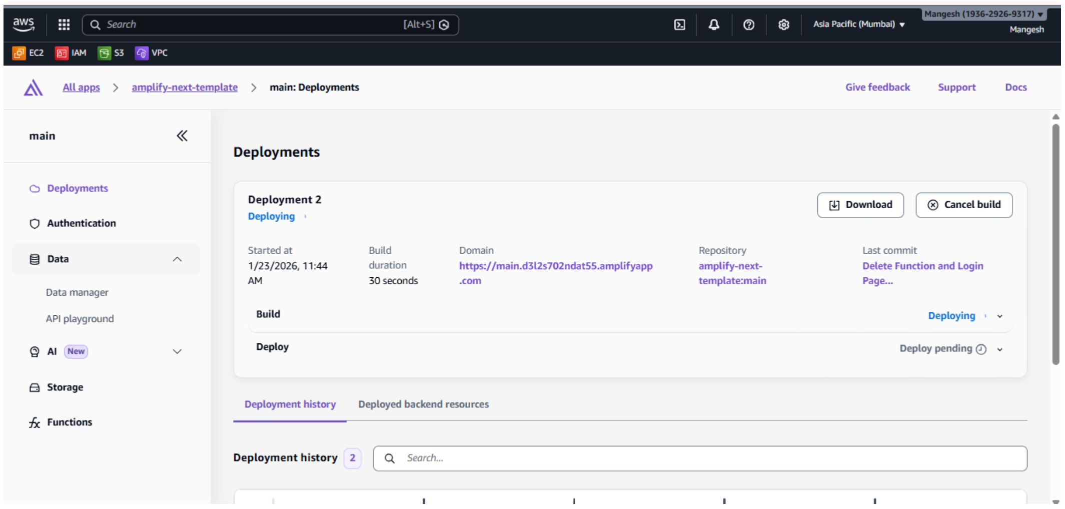
```
91937@Mangesh MINGW64 /c/AWS Git/amplify-next-template (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   app/layout.tsx
        modified:   app/page.tsx
        modified:   next-env.d.ts
        modified:   package-lock.json
        modified:   package.json
        modified:   tsconfig.json

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        app/AuthenticatorWrapper.tsx

no changes added to commit (use "git add" and/or "git commit -a")

91937@Mangesh MINGW64 /c/AWS Git/amplify-next-template (main)
```

Simply add the changes use → git add* command and them commit the changes use command → git commit -m " Note"



```
91937@Mangesh MINGW64 /c/AWS Git/amplify-next-template (main)
$ git add *
warning: in the working copy of 'tsconfig.json', LF will be replaced by CRLF the next time Git touches it
The following paths are ignored by one of your .gitignore files:
amplify_outputs.json
node_modules
hint: Use -f if you really want to add them.
hint: Disable this message with "git config set advice.addIgnoredFile false"

91937@Mangesh MINGW64 /c/AWS Git/amplify-next-template (main)
$ git commit -m "Delete Function and Login Page added "
[main 15c1537] Delete Function and Login Page added
 7 files changed, 38708 insertions(+), 16073 deletions(-)
 create mode 100644 app/AuthenticatorWrapper.tsx

91937@Mangesh MINGW64 /c/AWS Git/amplify-next-template (main)
$ git push
info: please complete authentication in your browser...
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 144.06 KiB | 3.06 MiB/s, done.
Total 10 (delta 8), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (8/8), completed with 8 local objects.
To https://github.com/mangeshbonde/amplify-next-template.git
```

Check the deployment status in AWS Amplify service.



Deployments

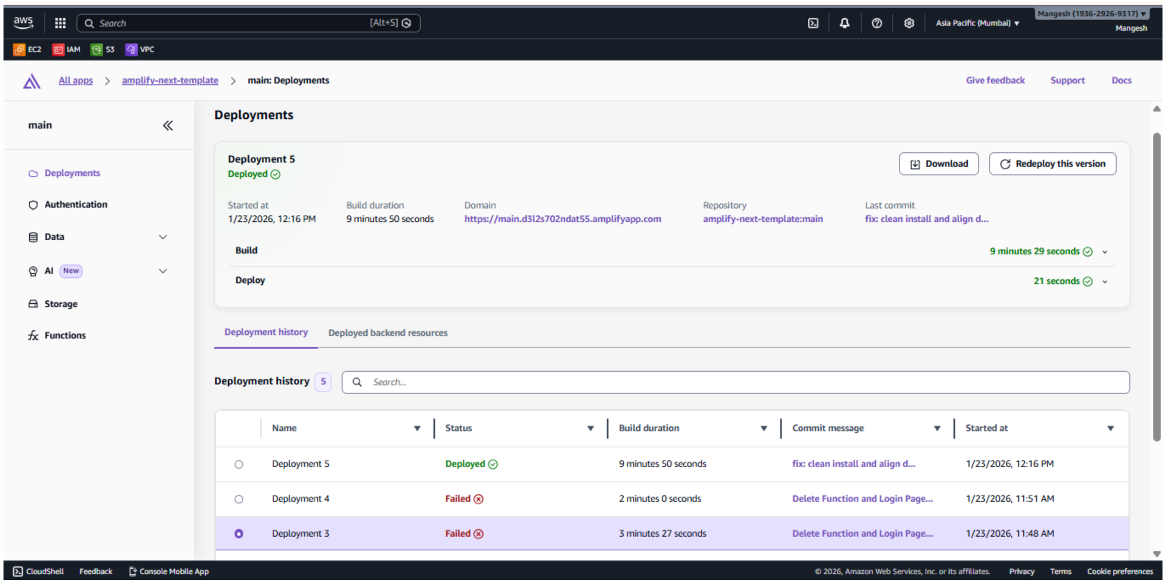
Deployment 2
Deploying

Started at	Build duration	Domain	Repository	Last commit
1/23/2026, 11:44 AM	30 seconds	https://main.d3l2s702ndat55.amplifyapp.com	amplify-next-template:main	Delete Function and Login Page...

Build — Deploying
Deploy — Deploy pending

Deployment history 2

We can see here project is successfully deployed after many unsuccessfull attempts.


aws
Search [Alt+S]
Asia Pacific (Mumbai)
Mangesh (1936-2926-9317)
Mangesh
EC2 IAM S3 VPC
All apps > amplify-next-template > main: Deployments
Give feedback Support Docs
main
Deployments
Authentication
Data
AI New
Storage
Functions
Download
Redeploy this version
Deployment history
Deployed backend resources
Search...
CloudShell Feedback Console Mobile App
© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Deployments

Deployment 5
Deployed

Started at	Build duration	Domain	Repository	Last commit
1/23/2026, 12:16 PM	9 minutes 50 seconds	https://main.d3l2s702ndat55.amplifyapp.com	amplify-next-template:main	fix: clean install and align d...

Build — 9 minutes 29 seconds
Deploy — 21 seconds

Deployment history 5

	Name	Status	Build duration	Commit message	Started at
○	Deployment 5	Deployed ✓	9 minutes 50 seconds	fix: clean install and align d...	1/23/2026, 12:16 PM
○	Deployment 4	Failed ⊗	2 minutes 0 seconds	Delete Function and Login Page...	1/23/2026, 11:51 AM
●	Deployment 3	Failed ⊗	3 minutes 27 seconds	Delete Function and Login Page...	1/23/2026, 11:48 AM

Make backend updates

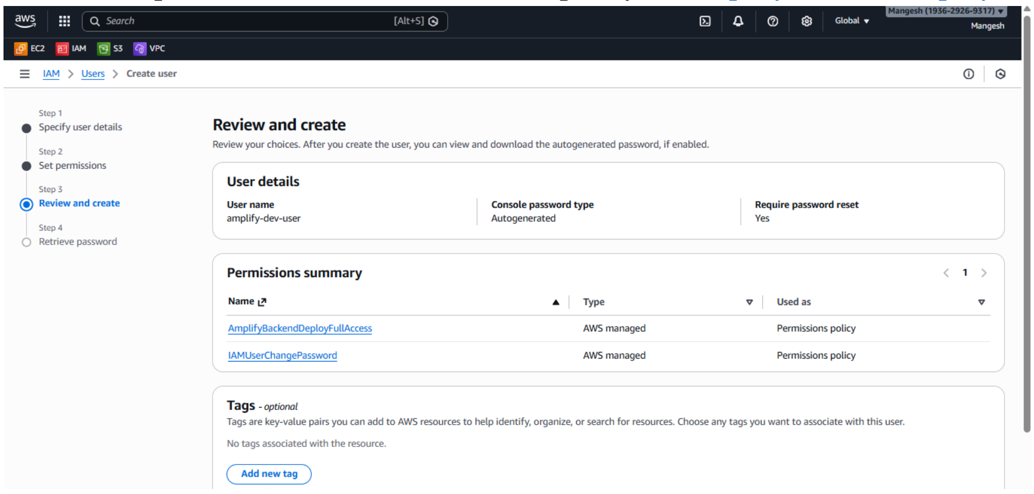
Let's update our backend to implement per-user authorization rules, allowing each user to only access their own to-dos.

Set up local AWS credentials

To make backend updates, we are going to require AWS credentials to deploy backend updates from our local machine.

Skip ahead to step 8, if you already have an AWS profile with credentials on your local machine, and your AWS profile has the AmplifyBackendDeployFullAccess permission policy.

For next step create iam user with attached policy of AmplifyBackendDeployFullAccess



Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name	Console password type	Require password reset
amplify-dev-user	Autogenerated	Yes

Permissions summary

Name	Type	Used as
AmplifyBackendDeployFullAccess	AWS managed	Permissions policy
IAMUserChangePassword	AWS managed	Permissions policy

Tags - *optional*

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

Create Access Keys (Credentials)

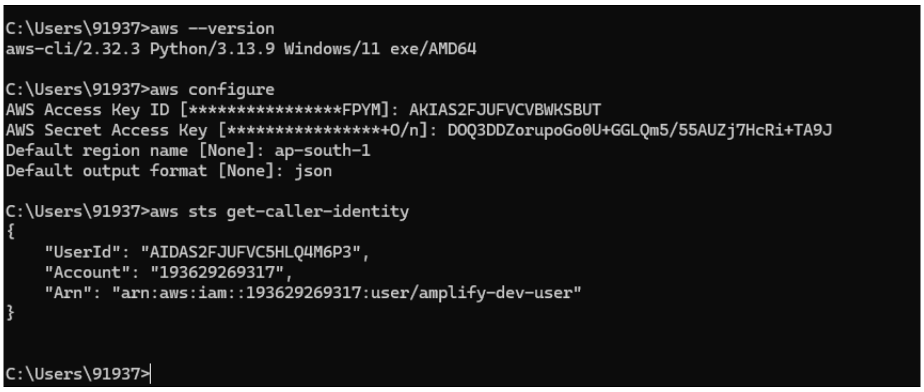
- Open the user amplify-dev-user
- Go to Security credentials tab
- Scroll to Access keys
- Click Create access key

Choose:

- Command Line Interface (CLI)

Confirm → Create access key

Configure in CLI

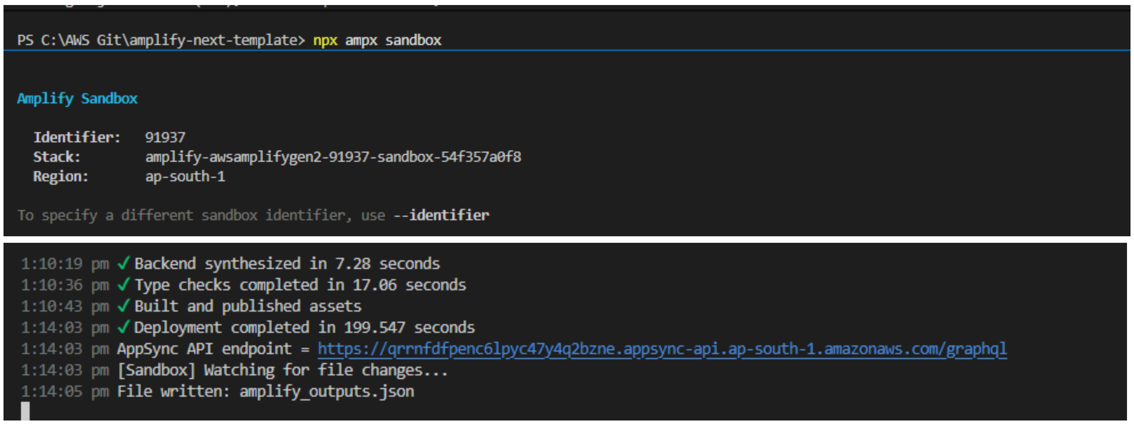


```
C:\Users\91937>aws --version
aws-cli/2.32.3 Python/3.13.9 Windows/11 exe/AMD64

C:\Users\91937>aws configure
AWS Access Key ID [****************FPYM]: AKIAS2FJUFVCVBWKSBUT
AWS Secret Access Key [****************+O/n]: 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
Default region name [None]: ap-south-1
Default output format [None]: json

C:\Users\91937>aws sts get-caller-identity
{
    "UserId": "AIDAS2FJUFVC5HLQ4M6P3",
    "Account": "193629269317",
    "Arn": "arn:aws:iam::193629269317:user/amplify-dev-user"
}


C:\Users\91937>
```

Deploy cloud sandbox

To update your backend without affecting the production branch, use Amplify's cloud sandbox. This feature provides a separate backend environment for each developer on a team, ideal for local development and testing.

To start your cloud sandbox, run the following command in a **new Terminal window**:

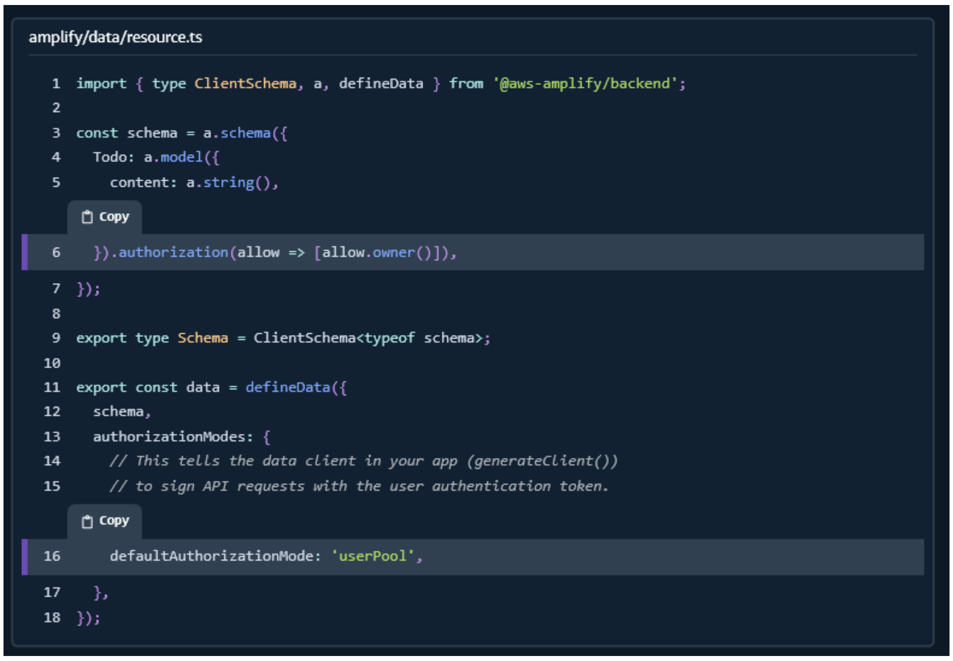


```
PS C:\AWS Git\amplify-next-template> npx ampx sandbox


Amplify Sandbox

  Identifier:    91937
  Stack:         amplify-awsamplifygen2-91937-sandbox-54f357a0f8
  Region:        ap-south-1

To specify a different sandbox identifier, use --identifier

1:10:19 pm ✓ Backend synthesized in 7.28 seconds
1:10:36 pm ✓ Type checks completed in 17.06 seconds
1:10:43 pm ✓ Built and published assets
1:14:03 pm ✓ Deployment completed in 199.547 seconds
1:14:03 pm AppSync API endpoint = https://qrrnfdfpenc6lpyc47y4q2bzne.appsync-api.ap-south-1.amazonaws.com/graphql
1:14:03 pm [Sandbox] Watching for file changes...
1:14:05 pm File written: amplify_outputs.json
```

Implement per-user authorization

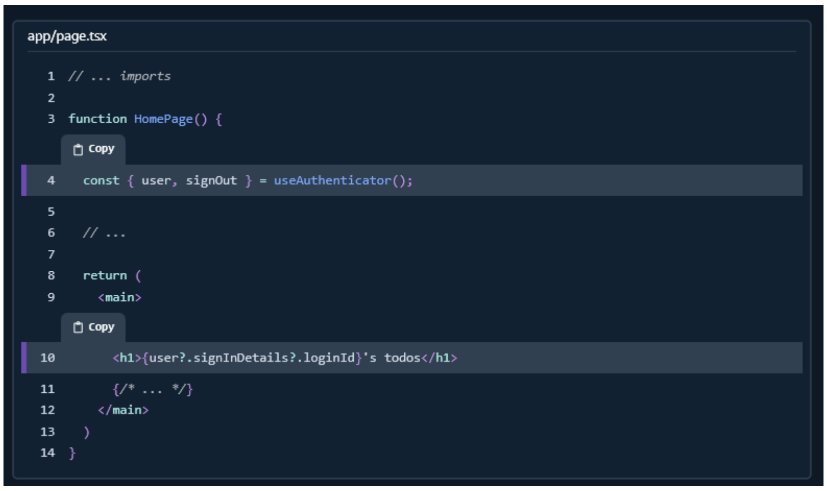
The to-do items in the starter are currently shared across all users, but, in most cases, you want data to be isolated on a per-user basis.

To isolate the data on a per-user basis, you can use an "owner-based authorization rule". Let's apply the owner-based authorization rule to your to-do items:


amplify/data/resource.ts

```
import { type ClientSchema, a, defineData } from '@aws-amplify/backend';

const schema = a.schema({
  Todo: a.model({
    content: a.string(),
  }).authorization(allow => [allow.owner()]),
});

export type Schema = ClientSchema<typeof schema>;

export const data = defineData({
  schema,
  authorizationModes: {
    // This tells the data client in your app (generateClient())
    // to sign API requests with the user authentication token.
    defaultAuthorizationMode: 'userPool',
  },
});
```

In the application client code, let's also render the username to distinguish different users once they're logged in. Go to your **app/page.tsx** file and render the user property from the useAuthenticator hook.


app/page.tsx

```
// ... imports

function HomePage() {
  const { user, signOut } = useAuthenticator();

  // ...

  return (
    <main>
      <h1>{user?.signInDetails?.loginId}'s todos</h1>
      {/* ... */}
    </main>
  )
}
```

What this does (simple words)

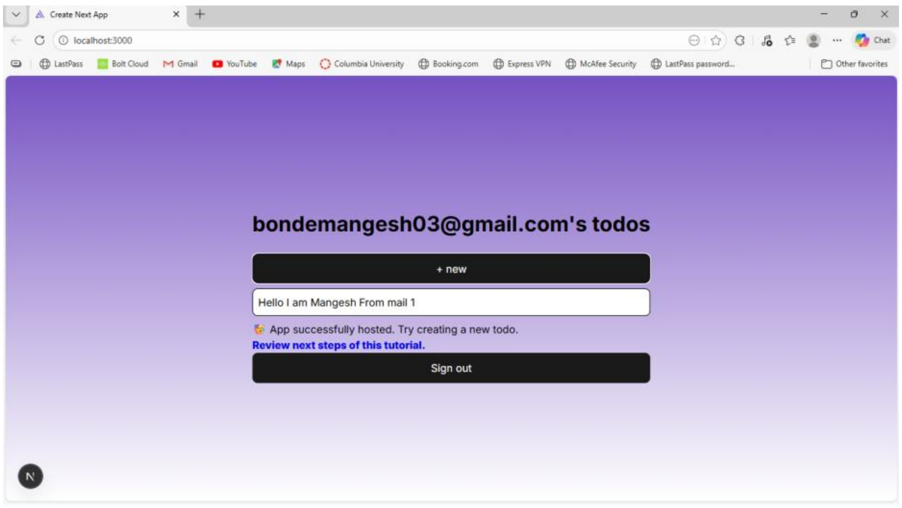
- Each todo belongs to **one user**
- Only owner can read/write/delete
- Auth token is required

Now, let's go back to your local application and test out the user isolation of the to-do items.

You will need to sign up new users again because now you're working with the cloud sandbox instead of your production backend.

After doing this steps simply save and run this on new terminal use npm run dev .

Here we create an account give email and password then put the otp sent on email then login .



bondemangesh03@gmail.com's todos

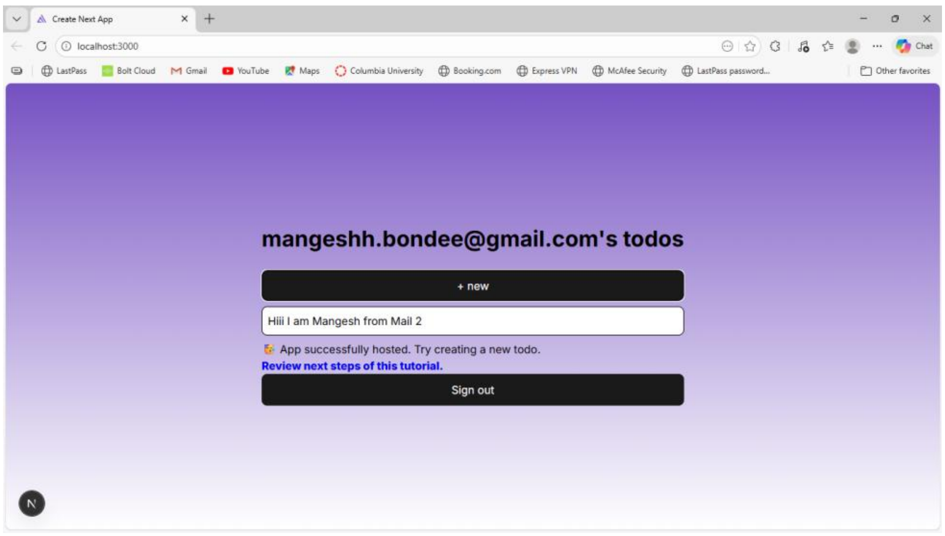
+ new

Hello I am Mangesh From mail 1

App successfully hosted. Try creating a new todo.
Review next steps of this tutorial.

Sign out

Create another account.



mangeshh.bondee@gmail.com's todos

+ new

Hiii I am Mangesh from Mail 2

App successfully hosted. Try creating a new todo.
Review next steps of this tutorial.

Sign out

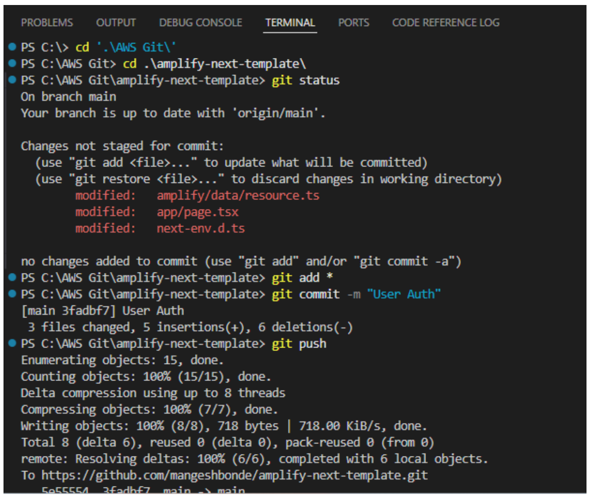
- Sign up again (new backend)
- Create todos
- Sign out
- Login with different user
- Todos will NOT be shared

To get these changes to the cloud, commit them to git and push the changes upstream.

Git add *

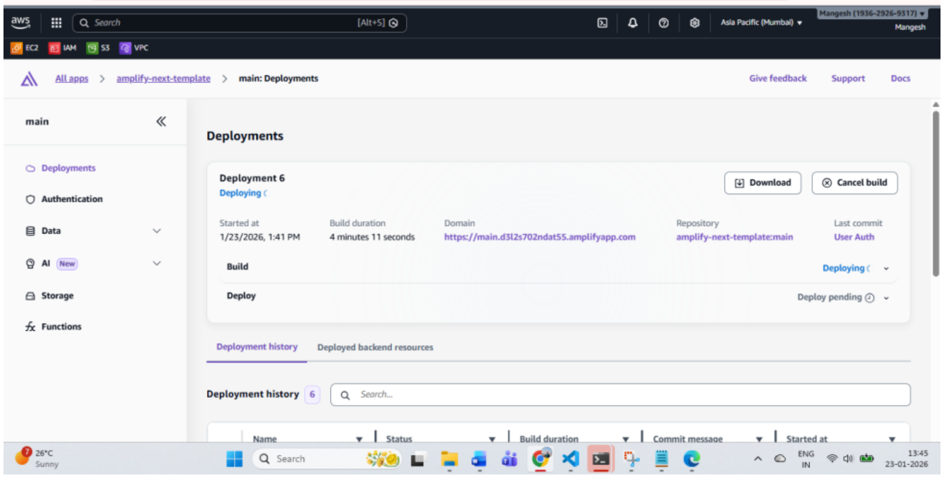
git commit -am "added per-user data isolation"

git push



```
PS C:\> cd '.\AWS Git\'
PS C:\AWS Git> cd .\amplify-next-template\
PS C:\AWS Git\amplify-next-template> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   amplify/data/resource.ts
        modified:   app/page.tsx
        modified:   next-env.d.ts

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\AWS Git\amplify-next-template> git add *
PS C:\AWS Git\amplify-next-template> git commit -m "User Auth"
[main 3fadbf7] User Auth
 3 files changed, 5 insertions(+), 6 deletions(-)
PS C:\AWS Git\amplify-next-template> git push
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 718 bytes | 718.00 KiB/s, done.
Total 8 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 6 local objects.
To https://github.com/mangeshbonde/amplify-next-template.git
   5e55554..3fadbf7  main -> main
```

We have done changes in main code and see it reflects in amplify service


All apps > amplify-next-template > main: Deployments

Deployments

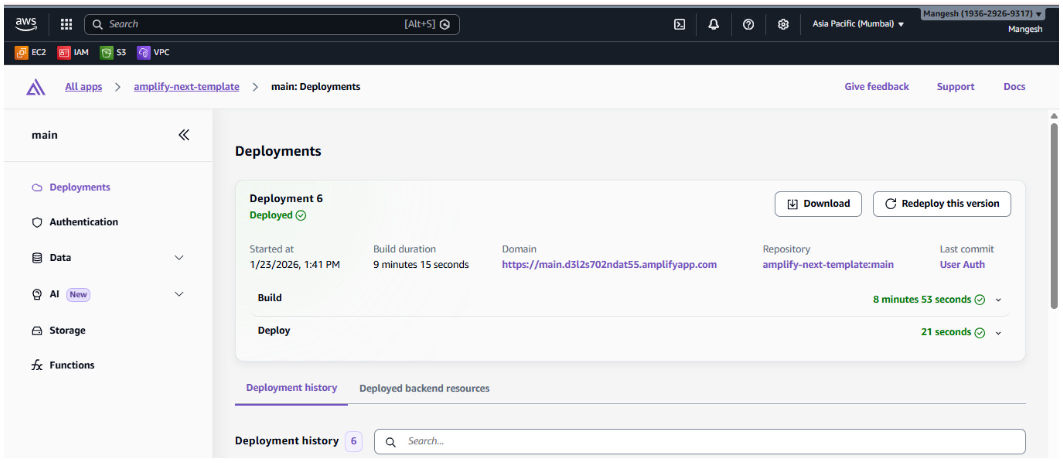
Deployment 6
Deploying

Started at	Build duration	Domain	Repository	Last commit
1/23/2026, 1:41 PM	4 minutes 11 seconds	https://main.d3l2s702ndat55.amplifyapp.com	amplify-next-template:main	User Auth

Build — Deploying
Deploy — Deploy pending

Deployment history 6

It takes upto 10 minutes to deploy.


All apps > amplify-next-template > main: Deployments

Deployments

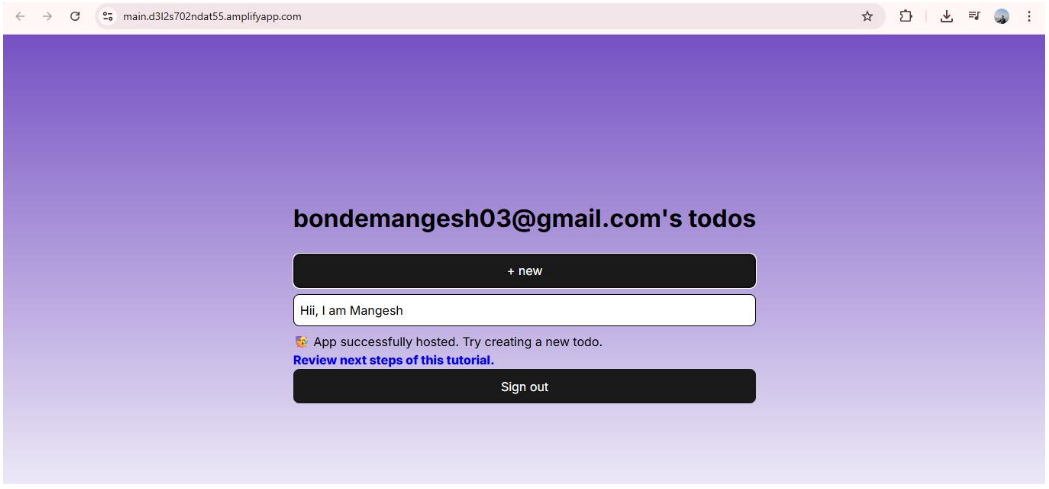
Deployment 6
Deployed

Started at	Build duration	Domain	Repository	Last commit
1/23/2026, 1:41 PM	9 minutes 15 seconds	https://main.d3l2s702ndat55.amplifyapp.com	amplify-next-template:main	User Auth

Build — 8 minutes 53 seconds
Deploy — 21 seconds

Deployment history 6

Lets check wheather the changes is reflected or not .


main.d3l2s702ndat55.amplifyapp.com

bondemangesh03@gmail.com's todos

+ new

Hii, I am Mangesh

🥳 App successfully hosted. Try creating a new todo.
Review next steps of this tutorial.

Sign out

Its works.

Key Learnings

- Cloud-native application design
- Serverless backend management
- CI/CD automation
- Secure authentication and authorization
- Debugging deployment issues

Conclusion

This project demonstrates how to build a modern cloud-enabled web application using Amplify and Next.js, leveraging serverless architecture and automated hosting.
You now understand how to integrate frontend, backend, and CI/CD workflows using AWS services.

References

- AWS Amplify Next.js Quickstart — https://docs.amplify.ws/nextjs/start/quickstart/nextjs-app-router-client-components/
- Amplify CLI Docs — https://docs.amplify.aws/cli
- Next.js App Router Docs — https://nextjs.org/docs

Note:

This project is implemented by following the official AWS Amplify and Next.js documentation step-by-step. AWS provides reference architectures and starter templates to demonstrate best practices for building cloud-native applications.
In real-world cloud development, engineers commonly use official documentation, sample templates, and reference guides to ensure:

- Correct architecture
- Security best practices
- Scalability and reliability
- Proper CI/CD implementation

All steps in this project—such as backend initialization, API creation, authentication, CI/CD setup, sandbox deployment, and authorization—are executed exactly as described in AWS official guides, and then tested, verified, and extended through local and cloud environments