## Qua. What Is Amplify? ?

Ans. => The Open-Source Amplify Provides The Following Products To Build Fullstack IOS, Android, Flutter, Web, And React Native Apps:

1. Amplify CLI - Configure All The Services Needed To Power Your Backend Through A Simple Command Line Interface.
2. Amplify Libraries - Use Case-Centric Client Libraries To Integrate Your App Code With A Backend Using Declarative Interfaces.
3. Amplify UI Components - UI Libraries For React, React Native, Angular, Vue And Flutter.\

The Amplify Hosting Is An AWS Service That Provides A Git-Based Workflow For Continuous Deployment & Hosting Of Fullstack Web Apps. Cloud Resources Created By The Amplify CLI Are Also Visible In The Amplify Console. .

## Qua. How To Install And Configure The Amplify CLI ?

Ans. =>
Video Link

Steps

1. Npm Install -G @Aws-Amplify/Cli

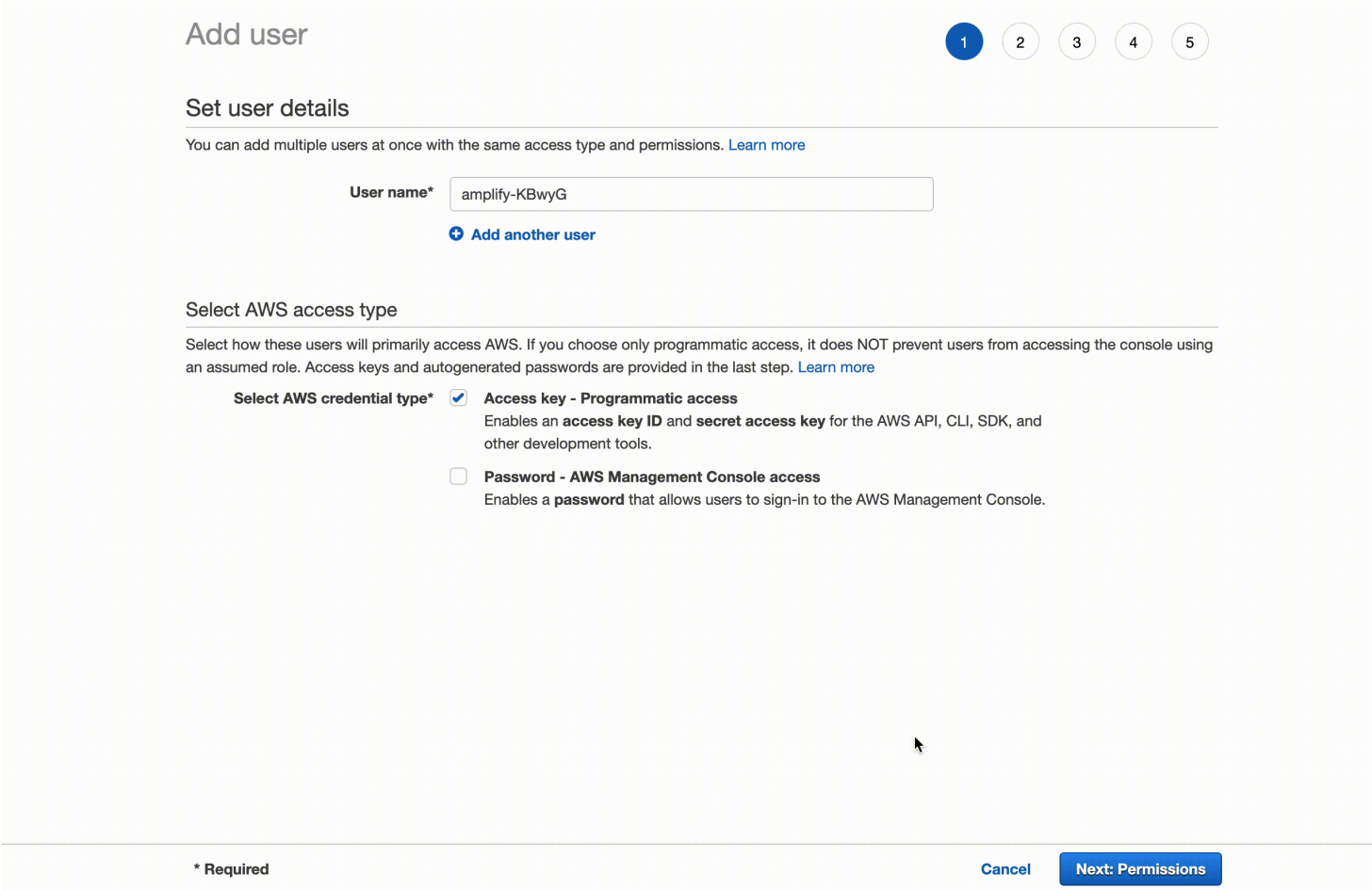
Now It's Time To Setup The Amplify CLI. Configure Amplify By Running The Following Command:

2. Amplify Configure

Once You're Signed In, Amplify CLI Will Ask You To Create An IAM User.

Specify The AWS Region
? Region:
Specify The Username Of The New IAM User:
? User Name:
Complete The User Creation Using The AWS Console:

Follow The And Create A User With AdministratorAccess-Amplify To Your Account To Provision AWS Resources For You Like AppSync, Cognito Etc.




Amplify CLI Will Then Ask You To Copy And Paste The AccessKeyId And The SecretAccessKey From Your Newly Created IAM User To Connect With Amplify CLI.

Enter The Access Key Of The Newly Created User:
? AccessKeyId:
? SecretAccessKey:
This Would Update/Create The AWS Profile In Your Local Machine
? Profile Name: Default
Successfully Set Up The New User.

In The Next Section, You'll Set Up The App And Initialize Amplify.

3. Create A New React App

To Get Started, First Create A New React App, And Then Install And Use The Amplify CLI To Start Adding Backend Capabilities To Your App.

From Your Projects Directory, Run The Following Commands:

```
Npx Create-React-App React-Amplified
Cd React-Amplified
```

This Creates A New React App In A Directory Called React-Amplified And Then Switches Into The New Directory.

From The React-Amplified Directory, Run The App By Using The Following Command:

```
Npm Start
```

4. Initialize A New Backend

With The App Running, It's Time To Set Up Amplify And Create The Backend Services To Support The App.

Open A New Terminal. From The React-Amplified Directory, Run:

```
Amplify Init
```

When You Initialize Amplify You'll Be Prompted For Some Information About The App, With The Option To Accept Recommended Values:

```
? Enter A Name For The Project Reactamplified
The Following Configuration Will Be Applied:

?Project Information
| Name:  Reactamplified
| Environment: Dev
| Default Editor: Visual Studio Code
| App Type: Javascript
| Javascript Framework: React
| Source Directory Path: Src
| Distribution Directory Path: Build
| Build Command: Npm Run-Script Build
| Start Command: Npm Run-Script Start
? Initialize The Project With The Above Configuration? Yes
Using Default Provider  Awscloudformation
? Select The Authentication Method You Want To Use: AWS Profile

...

? Please Choose The Profile You Want To Use Default
```

When You Initialize A New Amplify Project, A Few Things Happen:

1. It Creates A Top Level Directory Called Amplify That Stores Your Backend Definition. During The Tutorial You'll Add Capabilities Such As A GraphQL API And Authentication. As You Add Features, The Amplify Folder Will Grow With Infrastructure-As-Code Templates That Define Your Backend Stack.
2. 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.
3. It Modifies The .Gitignore File, Adding Some Generated Files To The Ignore List
4. 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
The Aws-Amplify Package Is The Main Library For Working With Amplify Libraries In Your Projects:

```
Npm Install Aws-Amplify
```

5. Set Up Frontend

Next, Configure Amplify So It Can Interact With Backend Services.

```
Import { Amplify } From 'Aws-Amplify';
Import AwsExports From './Aws-Exports';
Amplify.Configure(AwsExports);
```

And That's All It Takes To Configure Amplify. As You Add Or Remove Categories And Make Updates To Your Backend Configuration Using The Amplify CLI, The Configuration In Aws-Exports.Js Will Update Automatically.

Now That Your React App Is Set Up And Amplify Is Initialized, You Can Add An API In The Next Step.

6. Connect API And Database To The App

Now That You've Created And Configured A React App And Initialized A New Amplify Project, You Can Add A Feature. The First Feature You Will Add Is An API.

The Amplify CLI Supports Creating And Interacting With Two Types Of API Categories: REST And GraphQL.

The API You Will Be Creating In This Step Is A GraphQL API Using AWS AppSync (A Managed GraphQL Service) And The Database Will Be Amazon DynamoDB (A NoSQL Database).

Create A GraphQL API And Database

```
Amplify Add Api
```

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).
The CLI Should Open This GraphQL Schema In Your Text Editor.

```
Type Todo @Model {
  Id: ID!
  Name: String!
  Description: String
}
```

The Schema Generated Is For A Todo App. You'll Notice A Directive On The Todo Type Of @Model. This Directive Is Part Of The GraphQL Transform Library Of Amplify.

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 (Todo 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.

Deploying The API
To Deploy This Backend, Run The Push Command:

```
Amplify Push
```

? Are You Sure You Want To Continue? Y

# You Will Be Walked Through The Following Questions For GraphQL Code Generation

? Do You Want To Generate Code For Your Newly Created GraphQL API? Y
? Choose The Code Generation Language Target: Javascript
? Enter The File Name Pattern Of Graphql Queries, Mutations And Subscriptions: Src/Graphql/**/*.Js
? Do You Want To Generate/Update All Possible GraphQL Operations - Queries, Mutations And Subscriptions? Y
? Enter Maximum Statement Depth [Increase From Default If Your Schema Is Deeply Nested]: 2

? Are You Sure You Want To Continue? Yes

...

? Do You Want To Generate Code For Your Newly Created GraphQL API Yes
? Choose The Code Generation Language Target Javascript
? Enter The File Name Pattern Of Graphql Queries, Mutations And Subscriptions Src/Graphql/**/*.Js
? Do You Want To Generate/Update All Possible GraphQL Operations - Queries, Mutations And Subscriptions Yes
? Enter Maximum Statement Depth [Increase From Default If Your Schema Is Deeply Nested] 2

Now The API Is Live And You Can Start Interacting With It!

The API You Have Deployed Is For A Todo App, Including Operations For Creating, Reading, Updating, Deleting, And Listing Todos.

Next, Run The Following Command To Check Amplify's Status:

```
Amplify Status
```

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 | Myapi | No Change | Awscloudformation |

To View The GraphQL API In The AppSync Console At Any Time, Run The Following Command:

Try Running A Couple Of Mutations Locally And Then Querying For The Todos:

```
Mutation CreateTodo {
  CreateTodo(Input: {
    Name: "Build An API"
    Description: "Build A Serverless API With Amplify And GraphQL"
  }) {
    Id
    Name
    Description
  }
}

Query ListTodos {
  ListTodos {
    Items {
      Id
      Description
      Name
    }
  }
}
```

# Connect Frontend To API

In This Section You Will Create A Way To List And Create Todos From The React Application. To Do This, You Will Create A Form With A Button To Create Todos As Well As A Way To Fetch And Render The List Of Todos.

```
Import React, { UseEffect, UseState } From 'React';
Import { Amplify, API, GraphqlOperation } From 'Aws-Amplify';
Import { CreateTodo } From './Graphql/Mutations';
Import { ListTodos } From './Graphql/Queries';

Import AwsExports From "./Aws-Exports";
Amplify.Configure(AwsExports);
Const InitialState = { Name: '', Description: '' }

Const App = () => {
  Const [FormState, SetFormState] = UseState(InitialState)
  Const [Todos, SetTodos] = UseState([])

  UseEffect(() => {
    FetchTodos()
  }, [])

  Function SetInput(Key, Value) {
    SetFormState({ ...FormState, [Key]: Value })
  }

  Async Function FetchTodos() {
    Try {
      Const TodoData = Await API.Graphql(GraphqlOperation(ListTodos))
      Const Todos = TodoData.Data.ListTodos.Items
      SetTodos(Todos)
    } Catch (Err) { Console.Log('Error Fetching Todos') }
  }

  Async Function AddTodo() {
    Try {
      If (!FormState.Name || !FormState.Description) Return
      Const Todo = { ...FormState }
      SetTodos([...Todos, Todo])
      SetFormState(InitialState)
      Await API.Graphql(GraphqlOperation(CreateTodo, {Input: Todo}))
    } Catch (Err) {
      Console.Log('Error Creating Todo:', Err)
    }
  }
  Return (
# Html Code Is Below
  )
}
Const Styles = {
  Container: { Width: 400, Margin: '0 Auto', Display: 'Flex', FlexDirection: 'Column', JustifyContent: 'Center', Padding: 20 },
  Todo: {  MarginBottom: 15 },
  Input: { Border: 'None', BackgroundColor: '#Ddd', MarginBottom: 10, Padding: 8, FontSize: 18 },
  TodoName: { FontSize: 20, FontWeight: 'Bold' },
  TodoDescription: { MarginBottom: 0 },
  Button: { BackgroundColor: 'Black', Color: 'White', Outline: 'None', FontSize: 18, Padding: '12px 0px' }
}

Export Default App
```

`

```
<Div Style={Styles.Container}>
  <H2>Amplify Todos</H2>
  <Input
    OnChange={Event => SetInput('Name', Event.Target.Value)}
    Style={Styles.Input}
    Value={FormState.Name}
    Placeholder="Name"
  />
  <Input
    OnChange={Event => SetInput('Description', Event.Target.Value)}
    Style={Styles.Input}
    Value={FormState.Description}
    Placeholder="Description"
  />
  <Button Style={Styles.Button} OnClick={AddTodo}>Create Todo</Button>
  {
    Todos.Map((Todo, Index) => (
      <Div Key={Todo.Id ? Todo.Id : Index} Style={Styles.Todo}>
        <P Style={Styles.TodoName}>{Todo.Name}</P>
        <P Style={Styles.TodoDescription}>{Todo.Description}</P>
      </Div>
    ))
  }
</Div>
```

`

# Add Authentication

Authentication With Amplify

Amplify Uses Amazon 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
To Add Authentication To Your App, Run This Command:

```
Amplify Add Auth
```

To Deploy The Service, Run The Push Command:

```
Amplify Push
```

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
```