Building a Serverless Web Application

with AWS Lambda, Amazon API Gateway, AWS Amplify, Amazon DynamoDB, and Amazon Cognito

Presents By

Barish S

Amazon Web Services(AWS)

- AWS allows us to easily scale our resources up or down as our needs change, helping us to save money and ensure that your application always has the resources it needs.
- AWS provides a highly reliable and secure infrastructure, with multiple data centers and a commitment to 99.99% availability for many of its services.
- AWS offers a wide range of services and tools that can be easily combined to build and deploy a variety of applications, making it highly flexible.
- AWS offers a pay-as-you-go pricing model, allowing us to only pay for the resources we actually use and avoid upfront costs and long-term commitments.

ABOUT THIS PROJECT

- It is a simple serverless web application that enables users to request unicorn rides from the Wild Rydes feet.

- The application will present users with an HTML-based user interface for indicating the location where they would like to be picked up and will interact with a RESTful web service on the backend to submit the request and dispatch a nearby unicorn.

- The application will also provide facilities for users to register with the service and log in before requesting rides.

AWS Applications (In this Project)

AWS CodeCommit

AWS Amplify

Amazon Cognito

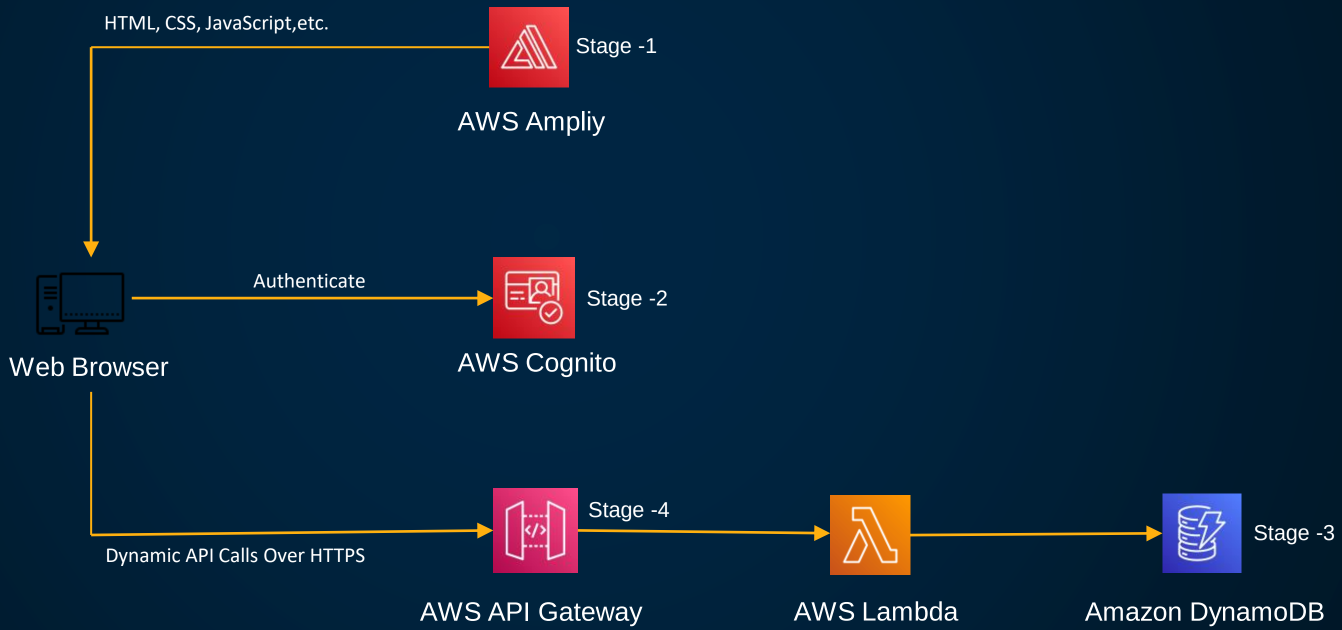
Amazon DynamoDB

AWS Lambda

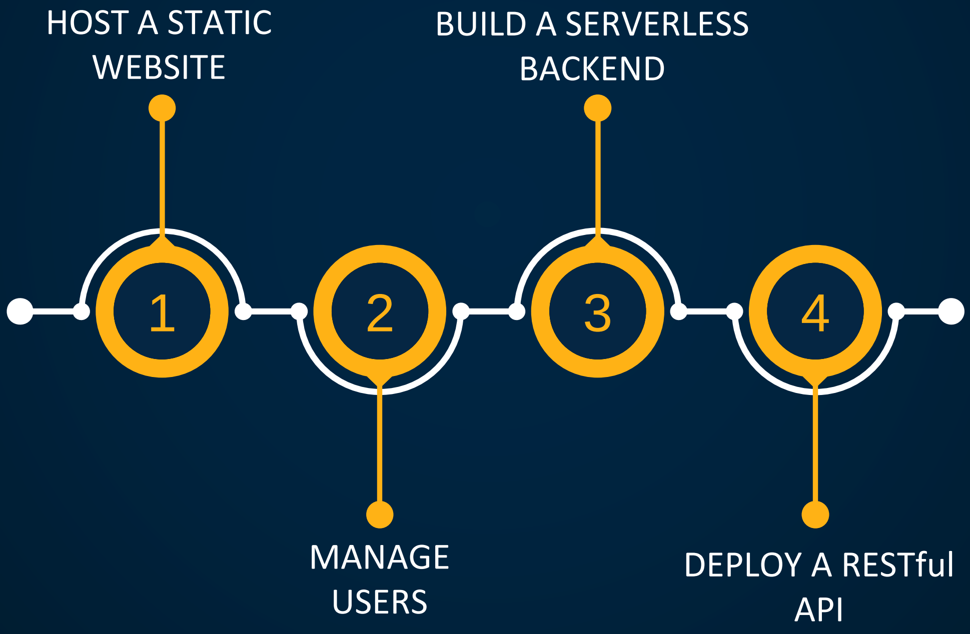
Amazon API Gateway

Cloud9

APPLICATION ARCHITECTURE



MODULES


HOST A STATIC WEBSITE
BUILD A SERVERLESS BACKEND
1
2
3
4
MANAGE USERS
DEPLOY A RESTful API

1. HOST A STATIC WEBSITE

- In this module, we are configure AWS Amplify to host the static resources for your web application with continuous deployment built in. The Amplify Console provides a git-based workflow for continuous deployment and hosting of full-stack web apps. In subsequent modules, you will add dynamic functionality to these pages using JavaScript to call remote RESTful APIs built with AWS Lambda and Amazon API Gateway.

AWS Amplify

- Amplify is a development framework and a web hosting service to build and deploy mobile and web applications on AWS.

- It also integrates with many AWS Services like AWS Cognito, AWS Lambda and Amazon S3.

Top Features

- ANALYSIS
- AUTHENTICATION
- DATA STORAGE

Amplify Benefits

- Pay per request and compute time.
- Integrates with the whole AWS Services and also with many programming languages

STEPS TO FOLLOW FOR HOST A STATIC WEBSITE

STEP_1: CREATE A GIT REPOSITORY

STEP_2: POPULATE THE GIT REPOSITORY

STEP_3: ENABLE WEB HOSTING WITH THE AWS AMPLIFY CONSOLE

2. MANAGE USERS

- In this module we are create an Amazon Cognito user pool to manage your users' accounts. You'll deploy pages that enable customers to register as a new user, verify their email address, and sign into the site.

Amazon Cognito

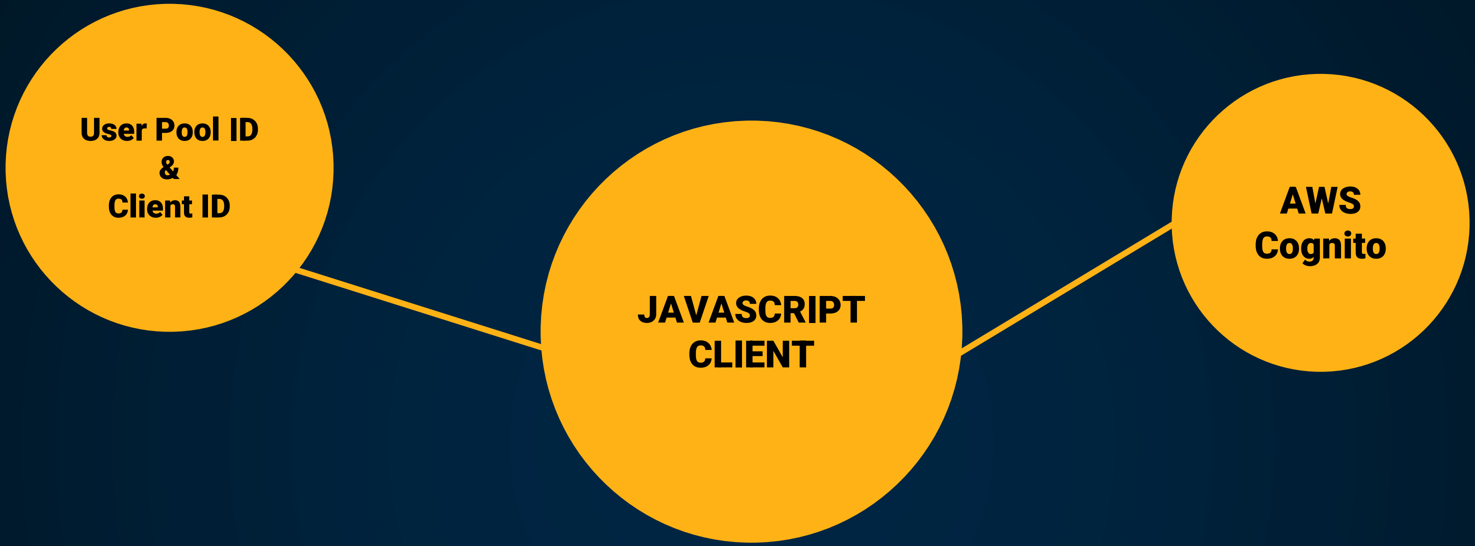
- It is an Amazon webservices product that controls user Authentication and Access for mobile application on internet-connected devices.

We can use it by Three Methods:

1. User pools. (In This project we use user pool) For user profile, what kind of attribute we need, can be customize.

2. Federated

3. Sync



- If the user is first time registering with email and password, aws congito send an OTP to the registered email to verify.
- The user will confirm the verification by entering the verify OTP and registered email, so that the user is Authenticated.

STEPS TO FOLLOW FOR MANAGE USER

STEP_1: CREATE AN COGNITO USER POOL AN INTEGRATE AN APP WITH USER POOL

STEP_2: UPDATE THE WEBSITE CONFIG FILE

STEP_3: VALIDATE THE IMPLEMENTATION

3. SERVERLESS SERVICE BACK-END

- In this module, We are use AWS Lambda and Amazon DynamoDB to build a backend process for handling requests for your web application. The browser application that you deployed in the first module allows users to request that a unicorn be sent to a location of their choice. To fulfill those requests, the JavaScript running in the browser will need to invoke a service running in the cloud.

AWS Lambda

- AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources, accelerating your journey from an idea to a modern, production application

ADVANTAGES:

1. Don’t need to worry about storage
2. Scaleup and scaledown is automated
3. Lambda save all the code to the Amazon S3 ,by encrypt .so that the data are secure

DO YOU THINK EC2 & LAMBDA ARE SAME ?

EC2 :

- Program should be change in server whenever we are updating
- Manual storage allocation,memory,OS,Database, CPU power.

LAMBDA :

- It can scaleup automatically based on the event
- Dont need to worry about storage allocation,memory,OS,Database, CPU power.

Amazon DynamoDB

- Amazon DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale. DynamoDB offers built-in security, continuous backups, automated multi-Region replication, in-memory caching, and data import and export tools.

ADVANTAGES:

1. Built-in security
2. Continuous backups, data import and export tools

USES CASES:

1. Snapchat
2. Scale gaming platforms(PUBG)

STEPS TO FOLLOW FOR SERVERLESS BACK-END

STEP_1: CREATE AN AMAZON DynamoDB TABLE

STEP_2: CREATE AN IAM ROLE FOR LAMBDA FUNCTION

STEP_3: CREATE A LAMBDA FUNCTION FOR HANDLING REQUEST

STEP_4: VALIDATE IMPLEMENTATION

4. DEPLOY A RESTFUL API

- In this module, We are use Amazon API Gateway to expose the Lambda function you built in the previous module as a RESTful API. This API will be accessible on the public Internet. It will be secured using the Amazon Cognito user pool you created in the previous module. Using this configuration, you will then turn your statically hosted website into a dynamic web application by adding client-side JavaScript that makes AJAX calls to the exposed APIs.

Amazon API Gateway

- APIs at any scale. APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services.

It Works on Two Types:

1. Monolithic Server Architecture
2. Microservice Architecture

- In this project we use API Gateway for second method.

STEPS TO FOLLOW FOR SERVERLESS BACK-END

STEP_1: CREATE A NEW REST API

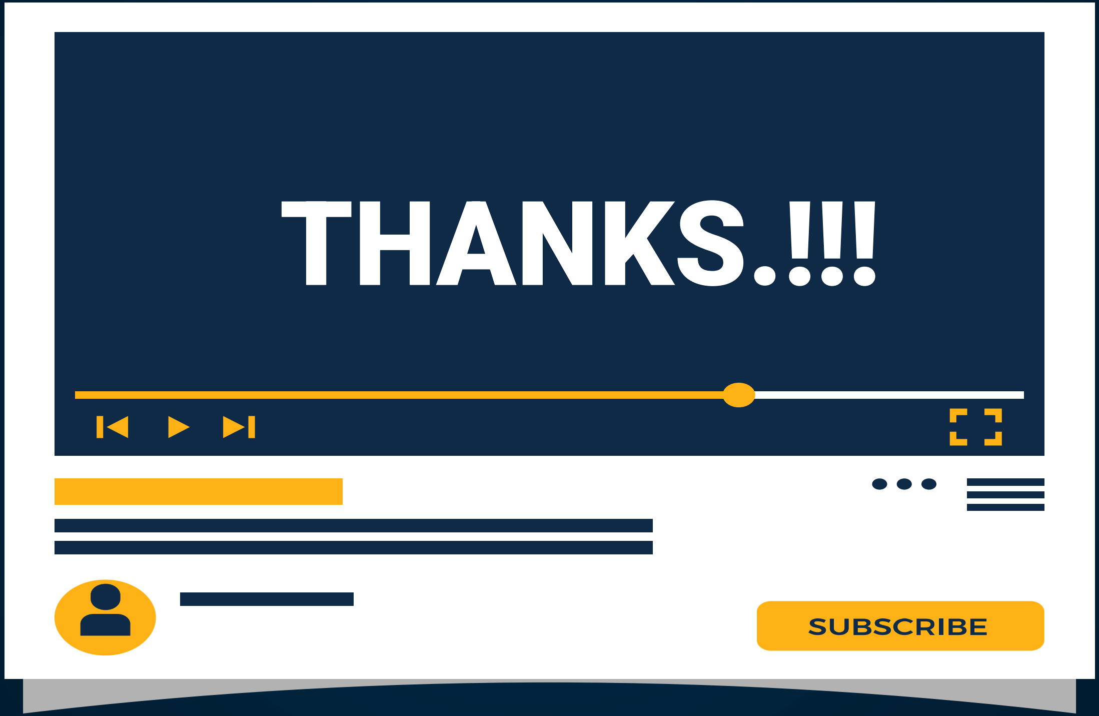
STEP_2: CREATE AUTHORIZER

STEP_3: CREATE A NEW SOURCE AND METHOD

STEP_4: DEPLOY API

STEP_5: UPDATE THE WEB CONFIG

STEP_6: VALIDATE IMPLEMENTATION


THANKS.!!!
SUBSCRIBE