© May 2021| IJIRT | Volume 7 Issue 12 | ISSN: 2349-6002

TANDOM: A Serverless SPA Using React and AWS

Aparajita Gogoi[1], Mr. Vignesh S.[2]
[1]*MCA Scholar, School of CS & IT, Dept of MCA, Jain (Deemed-to-be) University, Bangalore, India.*
[2]*Assistant Professor, School of CS & IT, Dept of MCA, Jain (Deemed-to-be) University, Bangalore, India*

Abstract - **Serverless computing has gained prominence in being the most compelling archetype in the deployment of various services and applications. Of late there has been a tremendous change in cloud technologies such as programming models, services and abstractions which is brought about through implementing serverless computing architectures in different products by cloud practitioners such as software application architects, or developers and end users. Since it is a relatively new technology and also due to its high architectural effect, its acquisition is lacking quite behind, and it is not being utilized to its maximum capacity as it should have been. And as such most developers or engineers might be aware that now cloud computing is not synonymous to only backend developments and devOps but due to rise in backend-as-a-service, the barrier to entry into cloud computing has been significantly lowered for frontend developers. In this project we will try bridging the gap between frontend and backend development by leveraging a new generation of services and tools provided by the AWS Amplify Framework, we will verify how such a serverless application can be adopted in running a business like an "hotel app" rather than using traditional web technologies. This serverless technology gives us a new opportunity to govern the reduction of some amount of operational costs by efficient optimization and management of cloud resources, or managing scalable cloud applications or entire development stack, one such ecosystem is that of React and Aws. We shall be using React to develop the front-end part of the website while the backend will be taken care of by using AWS Amplify services and other Aws services like DynamoDB or Lambda to maintain the database or some API's.**

Index Terms - **Amplify, AWS, Dynamo, Lambda, SPA**

I.INTRODUCTION

Serverless computing permits us to fabricate and run various types of applications and services without even having to contemplate about servers. In serverless computing, servers are still present, but the key difference is that AWS will handle all the server management.

AWS Lambda has acquired a ton of fame all throughout the most recent couple of years, with numerous architects/developers embracing it to function microservices and APIs for a negligible part of the expenses of traditionally facilitated hosted systems with predominant scaling abilities. Nowadays we can run fully functional Node.js web applications with React frontends on AWS Lambda without the requirement for any dedicated EC2 instances. With Lambda, architects are utilizing serverless infrastructure with Amazon functioning as a delegate between the client and code execution in a software containerization. environment, AWS Lambda does the following: Gives rules architects to be followed when submitting the code and how to enter the code into containers by means of automated measures, thus the whole backend development process becomes a well-managed service for the end users.

One of the popular front-end frameworks is React which can be implemented to create single-page applications (SPAs). React has recently emerged as one of the indispensable technologies upon which the building of the new and modern web development is being done on. In this project, we have tried to configure how a complete a web development environment in association with React is utilized to create an ecosystem which enables us in building applications. We went through all the vital and essential steps in getting our React app already which included deploying, connecting the frontend and the backend, and finally the react app being supported. Next, we went through as to how we can create certain React components to categorize and show our application. After that we configured how to add cloud services, for example like a database using JSON data and performed several checks in order to that ensure our code works on the cloud platform that we have selected. AWS and DynamoDB are interacted through the command line for example to create and use a DynamoDB table. Performing a lot of checks like how to create, load, and test a new role that can execute Lambda functions and interact with DynamoDB was part of this project. At last, we created and deployed an API gateway and concluded the project by configuring how to connect our React code to an API Gateway endpoint.

Software has become part of our daily lives now. Sadly, most organizations actually cannot deliver software effectively in the way that they are supposed to, significantly less do so as at the speed they are expected to remain serious. For the individuals who wish to keep up, not to mention lead, software delivery and its functionality should be drastically improved.

This is when serverless Architecture comes into the picture, it's based on cutting edge public cloud benefits that auto-scale and charge just when it is utilized. At the point when it scales, capacity planning, expense management are automatically done, the outcome is software that is simpler to set up, keep up, and mostly up to 99% less expensive.

However, Serverless Architectures are new and hence require a change by the way we recently pondered about architectures and work processes. Our objective was to build and work on a serverless application, in one basic, incredible, and exquisite experience.

II. PROBLEM STATEMENT

In conventional web environments, the proportion at which the computing power is squandered is extremely high. For instance, we were needed to pay to the service provider while paying little mind to the way that our servers were utilized or not. In any case, in the serverless infrastructure model, the valuing is determined dependent on the execution time and the occasions your functions are summoned. This can be bridged by utilizing serverless computing. With the ascent in managed and serverless administrations, it has gotten a lot simpler, for generally client-side designers to use these adaptable back-end services to fabricate the sorts of applications that, previously, would be out of their scope. Also, traditional servers are portrayed as a virtual machine tuning in for requests on the port number 80. For a long while now, this has been the norm of web development. Then again, the serverless model is one of many cloud computing alternatives which endeavours to remove the agony of scalability. Instead of running all day, every week a serverless application adopts an alternate strategy, it waits that as soon as requests comes in, it starts up as many instances according to the requirement depending on the situation to deal with the requests, closing down once the work is finished. Having said all this, in this project we will use different Aws services like Amplify and React JavaScript to build a practical single page serverless application on the cloud in order to find out how a serverless backend environment works and also how different is this from the traditional process.

III. LITERATURE REVIEW

Some of the methodologies used were:

1. For checking performance: Fast boot (Container cache, pre-warming, container optimization, Looking for other abstractions); Communications (Optimizing the storage server, Optimizing the communication path), Security issues: abstraction for function.
2. Configuring AWS Lambda for responding the notifications from the Auto Scaling Group: AWS Lambda function was configured in a way in which snapshot was automatically taken and then a new AWS EC2 instance was attached that was launched by the auto scaling group.
3. Survey of serverless commercial platforms like AWS Lambda, Alpha release, Google Cloud Functions, Microsoft Azure Functions, IBM OpenWhisk on the basis of different characteristics like Cost, performance, limits, security, deployment, accounting, programming languages, monitoring and debugging. Also benefits and drawbacks, workloads, frameworks etc.
4. Bottle necks and performance were determined on the basis of the following:
   - Serverless Computing
   - Performance Testing
   - Machine Learning and leveraging it for performance Validation.
5. An application called ExCamera, which uses cloud computing in addition with various workflows to alter, change, and encode recordings with low latency and cost was utilized to respond to the inquiries posed by conveying ExCamera

utilizing the IaaS model. They inferred that serverless registering model is on the ascent and is giving customary cloud foundation a run for their cash by dividing applications into short-running, progressively adaptable, and stateless capacities. However, serverless likewise presents new issues in both performance and security regions.

The following conclusions were made:

1. They concluded that serverless computing model is on the rise and is giving traditional cloud infrastructure a run for their money by splitting applications into short-running, dynamically scalable, and stateless functions. But serverless also introduces new problems in both performance and security areas.
2. The paper stated that serverless architectures can be considered as a new era of computation which can be adapted for use in much broader sense and there are possibilities of exploring more research avenues for the academic and research community in the arena of serverless computing.
3. The Function-as-a-Service (FaaS) model lends itself well to a number of common distributed application patterns, including compute-intensive there a wide variety of problems in this new technology ranging from infrastructure issues such as optimizations to the cold start problem to the design of a composable programming model.
4. The solution to improve performance-based issues/ bottlenecks and also to be able to measure is to use different kinds of artificial intelligence and machine learning for performance engineering.
5. Fine billing granularity, control and insight, the ability to run very fine arbitrary functions on request are offered by the current serverless technologies to its users however the drawback is that it works just on a few chosen applications.

IV. PROPOSED SYSTEM

React has recently emerged as one of the indispensable technologies upon which the building of the new and modern web development is being done on. In this project, we have tried to configure how a complete a web development environment in association with React is utilized to create a ecosystem which enables us in building applications. We went through all the vital and essential steps in getting our React app already which included deploying, connecting the frontend and the backend, and finally the react app being supported. Next, we went through as to how we can create certain React components to categorize and show our application. After that we configured how to add cloud services, for example like a database using JSON data and performed several checks in order to that ensure our code works on the cloud platform that we have selected. AWS and DynamoDB are interacted through the command line for example to create and use a DynamoDB table. Performing a lot of checks like how to create, load, and test a new role that can execute Lambda functions and interact with DynamoDB was part of this project. At last, we created and deployed an API gateway and concluded the project by configuring how to connect our React code to an API Gateway endpoint.

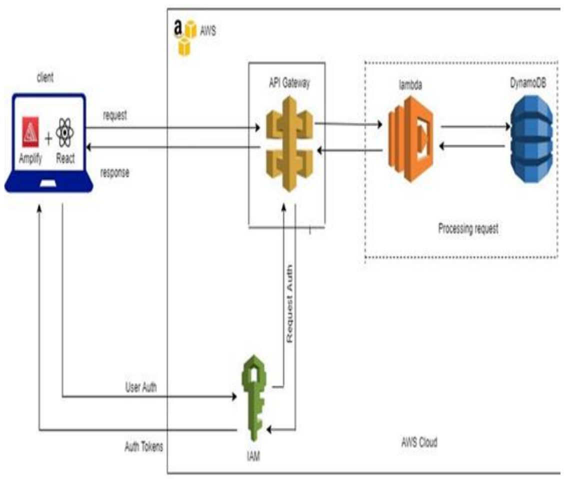


Fig 4.1: Proposed Architecture

V. METHODOLOGY

React has rapidly become the trendiest technology for front-end web and mobile development available. Along, with its noted use on sites like Facebook. And Netflix react offers an incredibly rich and flexible programming model, all built on JavaScript along with deep integration with cloud and serverless platforms, like, Amazon web services. The objective of this project is to show how to take a well, design HTML and CSS website converted into react, and then organize and optimize our entire react environment to manage versions, and deployments, and push it all to a cloud platform. We integrated with an API use, a NoSQL database and do all of it using well-established react best practices. React is at its heart, a front-end technology and while it does a lot of things, its primary purpose is to give you a programmatic method to build very rich, front-end interfaces and its only part of a larger ecosystem. Also, to learn a lot about other Technologies and both how to set them up on their own and then how to tie them into the react code.

1.React as the Front-End of an Ecosystem
Setting up our React environment: we're going to need a tool called npx. And if we have ever used the node package manager or npm, we may already have npx and familiarity the easiest way to find out if we have this tool is simply go out to a terminal and type in npx and we should see a long line of help text. The next step is to deploy our React application to check if all the dependencies have been correctly installed by using the command npm start by changing into the directory after which we connect our code to GitHub so that whenever we make changes it can be reflected in our repository too. Add AWS support with Amplify is important to connect to know what commit was happening, what did the code look like as amplify, this Is a great way to check that out, we now need to update our local application and deploy to AWS.

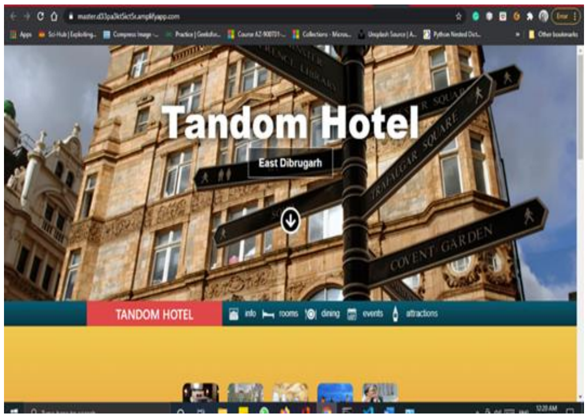
2.Build React Components to Render Structured Data
In this phase we develop our frontend that is basically building a site based on structured data after which we must organize your app by taking advantage of data-driven components using react. Next step is to build a menu based on dynamic data. At first, we will use JSON as a Mock for Dynamically Loaded Data (Links Json to simulate data in a database and then use that data dynamically in header. There are a number of other components that have similar cases, where we really should be using Dynamic data) after which we shall actually Load JSON as data in our React component. It is always a good practice to save our changes to GitHub.

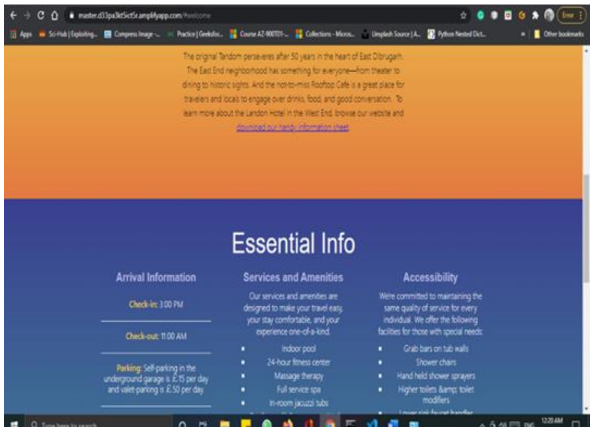
3.Using DynamoDB for simple data storage
So, at this stage, we will download and install the AWS command line interface. After that is completed, we need to do is create a user so that our CLI will use the command line interface to interact with AWS. In other words, this user will have the permissions given by the CLI. Now that we have the CLI and an IM user, we are ready to actually set up our database. For this, we need to log into AWS, we are going to call this user Dynamo user and it is going to need programmatic access. After the user creation is completed, we will get access ID, and secret, secret access key. Next step is to create DynamoDB tables for our components. We will take the help of the AWS console to Load data into DynamoDB, we can do this by two ways: Loading single-value data into DynamoDB programmatically and loading multi-value data into DynamoDB programmatically.

4. Load Dynamic Data into React Components
In this phase we create a role for Lambda and DynamoDB access and then Load DynamoDB data from a Lambda function, also perform a test to check our Lambda functions from AWS. After completion of that test, we write Lambda functions for all tables in our database. Next step is to create an API Gateway and finally create a GET endpoint for services, then deploy our API Gateway to a new stage. We must validate that our endpoint is dynamic and then add more endpoints to your REST API.

5. Connecting React to an API
In this phase we add a React Hook and a state variable, after that we create a side effect with the use Effect Hook. We now try to request data from an API Gateway and finally update CORS headers for our application. Our Tandom Hotel API calls are now finished.
We can now view the application we deployed into AWS using its services

V.RESULTS





master.d33pa3kt5ict5r.amplifyapp.com/#welcome

The original Tandom perseveres after 50 years in the heart of East Dibrugarh. The East End neighborhood has something for everyone—from theater to dining to historic sights. And the not-to-miss Rooftop Cafe is a great place for travelers and locals to engage over drinks, food, and good conversation. To learn more about the Landon Hotel in the West End, browse our website and download our handy information sheet.

Essential Info

Arrival Information

Check-in: 3:00 PM

Check-out: 11:00 AM

Parking: Self-parking in the underground garage is £15 per day and valet-parking is £50 per day.

Services and Amenities

Our services and amenities are designed to make your travel easy, your stay comfortable, and your experience one-of-a-kind.

- Indoor pool
- 24-hour fitness center
- Massage therapy
- Full service spa
- In-room jacuzzi tubs

Accessibility

We're committed to maintaining the same quality of service for every individual. We offer the following facilities for those with special needs:

- Grab bars on tub walls
- Shower chairs
- Hand held shower sprayers
- Higher toilets &amp; toilet modifiers

VI.CONCLUSION

After completion of the implementation, it was found that although cloud computing is a new concept it definitely has improved the entire web development right from deploying the website to maintaining the database or handling the servers. There were certain pros as well as cons. Some of the cons I observed were: Serverless empowers architectures which are event-based, which a many individuals might not be acquainted with. In addition to that, serverless is so relatively new that the tools that are currently available right now is moderately juvenile. It tends to be difficult to get things done as basic as stack traces. It has a heavier dependence on vendor ecosystems, so there's always that risk of vendor lock-in. Testing and debugging are often difficult in a serverless infrastructure. Also, Serverless infrastructures are not the right fit for long-running processes. On the positive side the few observations I made are: The server management is handled by the service provider. Engineers are only billed for the server space utilized hence reducing cost to a great extent. The possibility of quick deployments and updates make it very efficient. Code runs nearer to the end user client thus latency is often reduced.

REFERENCE

[1] Mingyu Wu, Zeyu Mi, and Yubin Xia (2020),” A Survey on Serverless Computing and its Implications for Joint Cloud Computing”, 2020 IEEE International Conference on Joint Cloud Computing (JCC)

[2] Dr. R. Arokia Paul Rajan, 2018, “Serverless Architecture-A Revolution in Cloud Computing “

[3] Ioana Baldini, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski, Philippe Suter, (2017), “Serverless Computing: Current Trends and Open Problems”, arXiv:1706.03178v1 [cs.DC] 10 Jun 2017

[4] Deepak Khatri, Sunil Kumar Khatri, Deepti Mishra, June 2020, “Potential Bottleneck and Measuring Performance of Serverless Computing: A Literature Study “, 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO) Amity University, Noida, India. June 4-5, 2020

[5] Van Eyk, E., Toader, L., Talluri, S., Versluis, L., Uta, A., & Iosup, A. (2018), “Serverless Is More: From PaaS to Present Cloud Computing”, Delft University of Technology and Vrije Universiteit Amsterdam

[6] Comparison of Serverless architectures, Accessed on 2nd August 2018,https://dzone.com/articles/4-use-cases-of-serverless-architecture

[7] https://martinfowler.com/articles/serverless.html

[8] https://www.lynda.com/Amazon-Web-Services-tutorials/Build-React-Application-Using-AWS-Amplify/2976141-2.html