IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 8, Issue 4, April 2019

Building a Serverless Progressive Web Application

Ajeeta Asthana[1], Akanksha Agrawal[2], Dr. Annapurna V.K[3]

Department of Computer Science and Engineering, The National Institute of Engineering, Mananthavadi Road, Mysuru, India[1,2]

Associate Professor, Department of Computer Science and Engineering, The National Institute of Engineering, Mananthavadi Road, Mysuru, India[3]

Abstract: Progressive web apps have the ability to incorporate the accessibility of the web with the features of the native apps. They provide cross-browser support and compatibility with mobile friendly frameworks. Serverless computing platform provide function as a service to end users while promising reduced high costs, high availability, fault tolerance, and dynamic elasticity for hosting micro services. It provides a platform for developers to execute the code in response to events without building and maintaining infrastructure. The third party apps or services would manage the server-side logic and state.

Keywords: Serverless Computing Architecture, Progressive Web Applications (PWAs), Native web apps, Amazon Web Services, Function as a Service (FAAS)

I. INTRODUCTION

Progressive web apps are web applications that load like regular web pages or websites. They can offer various functions to the end user such as working offline, push notifications and device hardware access, traditionally available only on native mobile applications. PWA's are more efficient than native apps. They work on-demand and are always accessible. They do not take up a smartphone's valuable memory or data. By choosing to use a PWA over a native version of the same application, users consume less data. PWA's are more economical from a developers perspective. They are faster to build and update. Serverless architecture refers to applications that are significantly dependent on third party services (known as Backend as a service, "BAAS") or on custom code that is run in ephemeral containers. The name "serverless computing" is used because the owner of the system does not have to purchase, rent or provision servers or virtual machines for the backend code to run on. Code optimizations help in increasing the speed of the app and also have a direct link for the reduction in operational costs. Serverless computing is an execution model where the cloud provider (AWS, Azure, Google Cloud) is responsible for executing a piece of code by dynamically allocating the resources. The code that is sent to the cloud provider is usually in the form of a function. Hence serverless is usually referred to as the "Function as a Service" or "FAAS". Scaling is completely automatic, elastic and managed by the provider. Serverless in its most simple form is an outsourcing solution. Setting up multiple environments for serverless is as easy as setting up a single environment.

II. NATIVE WEB APPS

Native Application is an application program that has been developed for use on a particular platform or device. They can interact with or take advantage of operating system features and other software that is typically installed on that platform, whether it be Android, iOS or windows phone. They use the developer tools provided by the operating system owner so that it's functionalities can be accessed. Developing a native web app can be time consuming and likely to require resources - money, developer time and other things. Moreover they require a download which means generating a considerable buy-in from customers first and losing the benefit of impulsive behaviour. The process of getting an application approved by the play store is long and tedious, updates also have to be play store approved to be featured there. Traditionally physical servers have been used to deploy back end of these native apps. The application runs in that server and developer is responsible for provisioning and managing the resources for it. There are a few issues with it . It is a developer's job to

- Keep the servers up even when they are not serving out any requests.
- Uptime and manage the servers and all its resources.
- Apply the appropriate security updates to the server.

As user download increase they need to manage scaling up the servers as well. Hence, manage scaling it down when there is not as much usage.

Servers need regular checking, updating and monitoring. It is likely that developers will need to undertake changes in users, permissions, email addresses and this may need detailed IT requirement.

III. METHODOLOGY

Building a high quality PWA has incredible benefits which helps in user engagements growth and increase in conversions. PWA's are built on the current web which is optimized for content distribution and commerce. There are several advantages of progressive web apps over the native application such as

- Low friction of distribution
- Mobile first approach
- Economical
- Book mark ability
- Large data savings

Every web page of a progressive web app is considered as a JavaScript site by Google. For PWA a new URL can be created and the GoogleBot will crawl it in the same manner it does for other pages on the web. As far as a service worker is considered, it is essentially a JavaScript file that runs separately from the main browser thread. It intercepts network requests, caching or retrieving resources from the cache and delivering push messages. These service workers only run over HTTPs.
Lighthouse is an open source and automated tool for improving the quality of our progressive web app,. It eliminates most of the manual testing that was previously required for native apps. Lighthouse is given a URL to audit, it runs multiple tests against the page to generate a report on how the page did. It has audits for performance, accessibility, progressive web apps and more.

A. Front-end using PWA:

Front-end consists of a basic note taking app built using ReactJS, a JavaScript library. To make it a progressive web app , Lighthouse is installed on Chrome Browser. This tool is used to measure the app, by creating audit reports of efficiency of different features and use it's feedback to upgrade the web app to a more efficient PWA level. A service worker is developed using JavaScript which is made to sit between the application and network. It will be used to intercept network requests and serve-up cached file, allowing users to work offline. Various progressive enhancements will be used according to user requirements like home screen capabilities, push notifications. etc.

B. Back-end using Serverless Deployment

Serverless is a cloud computing execution model where the cloud provider dynamically manages the allocation and provisioning of servers. A serverless application runs in a stateless compute container. These containers are event triggered, ephemeral and fully managed by the cloud provider. Function as a service (FAAS) is used to implement serverless architecture where engineers can deploy an individual function or a piece of business logic. A serverless back-end is set up consisting of creating database tables in MongoDB, S3 buckets for file uploads, authentication and authorization etc. Different serverless REST APIs are built to create notes, update notes, get, list and delete notes. Finally, followed by the deployment of APIs and testing them.

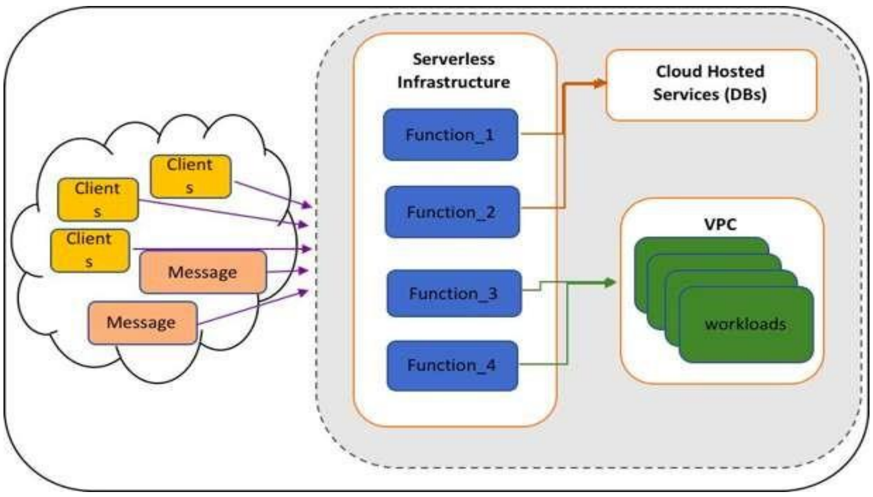

Fig 1 : Architecture of Backend

IV. IMPLEMENTATION

A service worker is a script that runs in the background of the browser. It is a network proxy used for handling network requests from a page. A service worker follows three steps during it's lifecycle. First step is to register the service worker in the main JavaScript code. Registration informs the browser of the location of service worker. Registration in followed by installation of the service worker in the background. Finally after successful installation step, the service worker transitions to the activation step. When PWA has access to the Internet, it sends GET requests to the serverless API and shows the content to the end user. This content persists in the local storage of the browser. When there is no Internet or poor network connection, contents are fetched from the local storage. In this case, when new requets are sent, it gets inserted in the request queue. As soon as the app returns to the online state, it calls requests present in the queue where all the requests are stored. The grid is updated again with a sync function. If there is any local ID data, it is either synced up pr updated with the server. This content is displayed by the PWA and also saved in the local storage of the application.

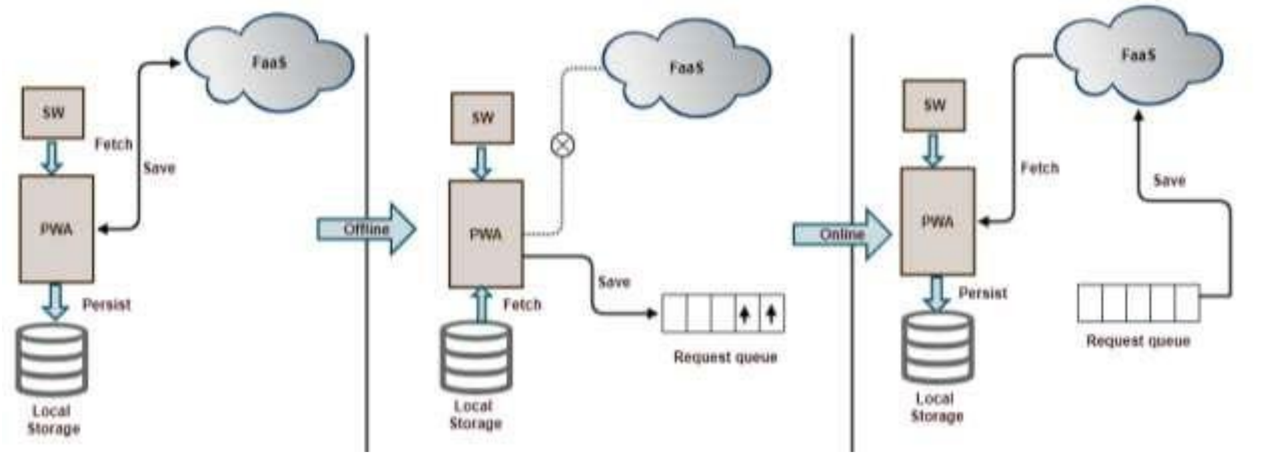

Fig 2 : Flow of Application

V. LIMITATIONS

One of the most inherent drawback of serverless architecture is the management of state. Stateless components interact with other stateful components to persist any information beyond their immediate lifespan. Most of the components of the application communicate with each other over an un-optimized channel which leads to increase in latency. Progressive web apps are not supported by legacy devices with outdated web browsers. PWA's are written in JavaScript, they consume more of the battery to interpret the code as compared to native apps written in Kotlin or Swift.

CONCLUSION

A progressive web application can be treated as a standard application on a device. The ability of it to be run from a uniform resource locator (URL) makes it easier for use on any device with a browser. Consumer electronics will increasingly use PWA either internally or as an interface. They are just one class of applications written using a language and associated tools. There is still a need for a variety of approaches to programming and software development. The growing adoption of serverless application warrants the evaluation of the platform quality. The development of new techniques maximize the technology's potential. The performance efficiency of the platform is high when high throughput is the key rather than low latency.

REFERENCES

[1]. B. Frankston, "Progressive Web Apps [Bits Versus Electrons]," in IEEE Consumer Electronics Magazine, vol. 7, no. 2, pp. 106-117, March 2018
[2]. P. Castro, V. Ishakian, V. Muthusamy and A. Slominski, "Serverless Programming (Function as a Service)," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, 2017, pp. 2658-2659.L. F. Albuquerque Jr., F. S. Ferraz, Rodrigo F. A. P. Oliveira, and Sergio M. L.Galdino, "Function-as-a-Service/Platform-as-a-Service: Towards a Comparative Study on FaaS and PaaS," 2017 ICSEA: The Twelfth International Conference on Software Engineering Advances, ISBN: 978-1-61208-590-6, Brazil, 2017.
[3]. McGrath, G., & Brenner, P. R. (2017). Serverless Computing: Design, Implementation, and Performance. 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW).
[4]. https://developers.google.com/web/getting-started-pwa/
[5]. https://aws.amazon.com/lambda/