Building Serverless Website on GitHub Pages

Prayudi Utomo¹*, Falahah²

¹Informatics Department, Universitas Widyatama, Bandung, Indonesia
²Information System Department, School of Industrial Engineering, Telkom University, Bandung, Indonesia

Email: *prayudi.utomo@gmail.com

Abstract. GitHub provides the environment for developing and publishing website in integrated environment. By GitHub Pages, we can host the static web easy, fast and free. It helps the developer for integrating development and deployment process, because GitHub Pages is integrating with GitHub environment that support continuous integration and continuous delivery, and also become a content delivery network (CDN), as a part of JAMstack building block. Unfortunately, the usage GitHub as web hosting and JAMstack as an approach for developing web is not very popular among developer in Indonesia. The purpose of this research is to show how implement the JAMstack approach on designing and building website and host it on the GitHub. The website we designed using JAMstack approach by implementing Hugo as static site generator (SSG) and GitHub as content delivery network (CDN). We also insert API for showing the easiness of integrating lots of components on JAMstack approach. We do several tests to check the quality of the website, such as functional test and performance test. The performance test results show that Google PageSpeed score is 97% (A) and YSlow score is 91% (A). It shows that the website can run in good performance although it connected with others services through API.

1. Introduction

Serverless computing provides some services that easy to access and implemented. Hosting a website in serverless environment is easier than conventional approach when the users should set up and configure the web server by themselves [1]. Serverless computing, or also popular as Function as a Services (FaaS), is predicted would be the next trend in cloud business [2]. The term of serverless is to describe the application that cloud-hosted and rely on the server on cloud to manage the server-side logic, and using ecosystem of cloud services such as database (Firebase, Parse) or authentication (such as Auth(), AWS incognito) [3]. The last services mentioned also popular as Backend as a Services (BaaS). Implementation of serverless computing can be done through some cloud providers such as AWS Lambda (Amazon), Google Cloud Functions, Azure Functions and IBM Cloud Functions [4].

GitHub, as a popular site among programmers to share their codes and repositories, also provides free services for web hosting [5].

Hosting application on serverless environment has been done by several researchers. Watson Developer Cloud can be used as platform for hosting chatbot application [6]. Pérez and Moltô has created the SCAR Framework for creating highly-parallel event-driven serverless applications. This framework can be run, and customized runtime environments defined as Docker images on top of AWS Lambda [7]. Hosting and running web applications in GitHub environment can increase productivity and improve quality of software development through automation and continuous integration. It can reduce effort of code reviewing process [8]. JAMStack is a new approach for website development. On
JAMStack approach, the architecture of web development is based on client-side JavaScript, reusable Application Programming Interface (API), and prebuilt Markup [9].

As we can conclude from previous researches, there is an emerging trend on using serverless approach as a platform for developing and hosting the application. Unfortunately, it is hard to find references or publications that describe the implementation of serverless approach in Indonesia. This condition gives us motivation to conduct a research how to develop and run an application using serverless approach. On this research we will build a web application using JAMstack architecture. We choose Hugo as a Static Site Generator and GitHub Pages as Content Delivery Network (CDN). As a study case, the website will provide the information about Kodein, a start-up IT company.

2. Methods

On this research, we will use JAMstack as an approach for building websites. One of the components of JAMstack architecture is Static Site Generator (SSG), as a tool for generating backbone of the website. There are many SSGs that we can choose. One of the popular SSGs among web developers is Hugo.

2.1. JAMstack

JAMstack is a new approach in building website and applications, which consist of three concepts [9]: JavaScript (to handle all dynamic programming during request/response, that running on the client), APIs (all server-side functions and database action are constructed as reusable APIs, and can be accessed over HTTPS with JavaScript), and MarkUp (the code or template of code should be pre-build at built time, using site generator or build tool).

The difference between JAMstack and traditional approach is shifted from application that mainly running on server-side into modular and client-side stacks. On JAMstack project, we can define some practical characteristics such as: entire site/app on a Content Delivery Network (CDN), atomic deploy, instant cache invalidation, everything lives in Git and automated builds. Figure 1 below shows the building block of website that using JAMstack paradigm, which are: Static Site Generator (SSG), content management system (CMS) and Content Delivery Network (CDN) [10]. Git also can be considered as CDN by utilized the Git repositories (GitHub, GitLab, and BitBucket) (CheonFong, 2019). For small and simple static website, we can use Hugo as static site generator [9].

![Figure 1. Building Block of JAMStack Website [10]](image_url)

JAMStack can be implemented in vary project, ranging from small site into complex web application that consist thousands of pages. Here is some type of JAMstack projects [9]: HTML Content, content from a CMS (especially headless CMS), web application, large web properties and hybrid experiences (web consists of variety contents such as main website, blog, documentation, and so on).
2.2. Hugo
Hugo is the oldest and popular framework for building the static website. Hugo is an open source Static Site Generator (SSG) that built on Go language (Go-lang) and also considered as a fast and flexible framework. The difference between SSG and CMS (Content Management Systems) like WordPress, is, CMS build a page every time the server receives the request from visitor. Meanwhile, SSG build a web page once when we create new content or edit it. Although it was written in Go-lang, implement Hugo does not need the knowledge of Go-lang if we want to implement Hugo. Hugo is a suitable tool for building this kind of websites: personal websites and blogs, non-technology business website, documentation website, and hybrid JAMstack based websites [11].

Implement the JAMstack Approach
The aim of this research is to build a website using JAMstack approach and implement it on serverless environment. As the case study, we choose to build the website for Kodein, a small start-up company that provide IT services. The steps are content design, JAMstack approach, and choose CDN to host the website.

2.3. Content Design
The website will show the information about Kodein. The content will consist of some basic information such as: company profile (vision, mission, value, and history), the service provided by company, the client’s profile, client’s testimonies, and blog for sharing information and knowledge.

The design pattern we choose for building the web site is Single Page Application (SPA) to support fast and easy access for the visitor. In SPA the browser will display all the content in one step and no page reload. Usually SPA using AJAX and HTML5 in building responsive website, and has some advantages such as: caching capabilities, linear user experience, fast and easy to debug with chrome [12]

2.4. JAMstack Approach
The site will be build using JAMstack approach as follow:

a. Static Site Generator: We choose Hugo as our tools to produce the static script for the website. Information Architecture of website will follow the Hugo structure and will consist of some files as shown in table 1.

b. Content layout: We choose Influencer template as content layout, which consist of some section as follow: title (left up), menu (top of the page), main page, content based on menu.

c. Application Programming Interface (API): API can be attached into JAMstack using 3 ways: direct access to remote API, an API Proxy, and using the build process [13]. On this website, we will use direct access to remote API. The API we will use here is the API to save and display the comment from visitor, a service from disqus.com. We can attach the API from disqus.com using 4 steps below:
1. Create account on disqus.com
2. Register the website address on disqus.com dashboard

| Folder name | Description | Contents in the folder |
|-------------|-------------|-----------------------|
| archetypes/ | The template for new content | *.md |
| Content/ | Store the content | Content/article, content/blog, etc |
| Data | Store the data | *.yaml, *.json, *.toml |
| Layouts | Store the template | *.html |
| Static | Store any static asset | *.css, *.js, and images |
| Themes | Store the themes used in the website | Content/data, content/static, etc |
| Config.toml | Contain the site settings | Configuration code |
3. Disqus will generate the script for embedding the form on our website. Copy the script and put on our website.
4. The script will generate comment form automatically and can accept the comment from visitor.
5. After visitor entering the comment, the web admin can read and reply the comment directly from the website. All comment is saved on disqus.com

Here is the step by step for attach the API and the scenario how the API works:
1. Create new <div> on blog index.html and insert the script below:

```html
<scr$</div>
<scr$ var disqus_config = function () {
this.page.url = PAGE_URL; // Replace PAGE_URL with your page's canonical URL variable
this.page.identifier = PAGE_IDENTIFIER; // Replace PAGE_IDENTIFIER with your page's unique identifier variable
};
(function() {
var d = document, s = d.createElement('script');
s.src = 'https://kodeinindonesia.disqus.com/embed.js';
s.setAttribute('data-timestamp', +new Date());
(d.head || d.body).appendChild(s);
})();
<noscript>Please enable JavaScript to view the comments powered by Disqus.</a>
</div>
```

2. Run the website, and try to insert some comment and reply. As we can see, the comment will be displayed on website directly (Figure 2), and the admin of the website can see the comment that saved on disqus.com (Figure 3).

![Figure 2. Displayed of Disqus API on website](https://kodeinindonesia.github.io/)

2.5. Content Delivery Network
We choose GitHub pages as CDN (content delivery network), based on some considerations as follow: its free, community based so it can support collaboration, and has versioning features to track the changes. We can host free static website on GitHub by creating an account and copy the file or upload the file (push file) using installed git component in our computer.
3. Results and Discussion
We upload the website into the GitHub pages and do some test such as functional test and performance test. The url of our website is https://kodeinindonesia.github.io/.

We also test the API comment both on the website or on disqus.com. Figure 4 show the comments and its reply on the web.

![Figure 3. Disqus dashboard](image)

![Figure 4. Using disqus API to handle the comments](image)

We do two types of testing which are functional testing (manual and automatic) and performance testing. Functional testing is conducted in manual to check all links and check the form on each page. We only check the links because our website does not have any forms. Automatic testing is running
using cookiemetrrix.com to check the cookie and Jigsaw (jigsaw.w3.org) to validate the HTML and CSS. Performance testing is running using Google PageSpeed test and YSlow test, that is conducted using GTMetrix.com. The result of automatic testing for cookiemetrrix.com and jigsaw show that the website is comply with Cookie law but still have some error in HTML/CSS. The result of performance test show that the website is running in good performance, which can be seen on figure 7. The score of Google PageSpeed is 97% and YSlow is 91%.

As we can see from the result above, GitHub can help developers to automate the deployment process by continuous integration and delivery. The utilization of GitHub Pages, as a part of GitHub feature, gives the developers the most practical ways for publishing the web application, that automatically connecting with deployment environment with less effort. Implementation of JAMstack approach on building website can produce a fast and reliable site and easy to scale up. API feature on JAMstack makes our web application can connect and communicate with many services on various platforms. The test results show our website can keep running on its good performance even though it connects with many services through API.

4. Conclusion
Serverless is a technology that make us easy for posting and managing the website. Recently we can host a serverless web page in very simple way and then we can manage the growth in the future. On this research we design, create and publish the website that was designed for IT software house and was develop by implementing JAMstack technology. On implementation of JAMStack, we divide the script on website based on Java (for dynamic script and calling API), API (on this website we use message services from disqus.com) and markup (the HTML itself). We choose the GitHub page as hosting services because it can support the integrated service for content delivery network (CDN) so the website can easily connect with API services (Disqus). We also use some automatic testing for measuring the performance and check the compliant of website to standard of HTML and CSS. The test shown the good result, except some error in HTML and CSS. By hosting the web page, we can show how to implement serverless technology by utilize services from GitHub and Disqus (as an API services). The website can be published easily without need any configuration setting from developer.

References
[1] Paakkunainen, O. 2019. Serverless computing and FaaS platform as a web application backend.
[2] Hellerstein, J.M., Faleiro, J.M., Gonzalez, J., Schleier-Smith, J., Sreekanti, V., Tumanov, A., & Wu, C. 2018. Serverless Computing: One Step Forward, Two Steps Back. ArXiv, abs/1812.03651.
[3] McGrath, M.G., & Brenner, P.R. 2017. Serverless Computing: Design, Implementation, and Performance. 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), pp.405-410.
[4] Malawski, M., Gajek, A., Zima, A., Balis, B., & Figiela, K. 2017. Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions.
[5] Cosentino, V., Izquierdo, J.L., & Cabot, J. 2017. A Systematic Mapping Study of Software Development with GitHub. IEEE Access, 5, pp.7173-7192.
[6] Yan, M., Castro, P.C., Cheng, P., & Ishakian, V. 2016. Building a Chatbot with Serverless Computing. MOTA ’16.
[7] Pérez, A., Moltó, G., Caballer, M., & Calatrava, A. 2018. Serverless computing for container-based architectures. Future Gener. Comput. Syst., 83, pp.50-59.
[8] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. 2015. Quality and productivity outcomes relating to continuous integration in GitHub. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering
(ESEC/FSE 2015).

[9] Biilmann, M. & Hawksworth, P. 2019. Modern Web development on the JAMstack: Modern
techniques for ultrafast sites and web applications, O'Reilly Media.

[10] Väänänen, J., 2019, JAMstack - Dynamically Functional Static Website, Theses, Oulu University
of Applied Sciences, Degree Programme in Information Technology, Software Development

[11] Karlík, R., 2019, Blogging platform utilizing Kentico Cloud and Jekyll static site generator, Master
Theses at Masaryk University Faculty of Informatics, Brno, Ceko.

[12] Joseph, R.J., 2015, Single Page Application and Canvas Drawing, International Journal of Web
& Semantic Technology (IJWesT), 6(1), January 2015

[13] Jain, A, 2019, Hugo in Action, Manning Publication,