PERSONAL TRAVEL BLOG ON IBM CLOUD STATIC WEB APPS

SUBMITTED BY

SABARISH.S

PHASE 4 SUBMISSION DOCUMENT

PHASE 4 : Development part 2

Topic : Continue building the travel blog by setting up the IBM Cloud Static Web App and deploying the website . Sign up for an IBM Cloud account . Create a new Static Web App and follow the prompts to set up the repository ,build pipeline and deployment options . Choose a static site generator like Jekyll or Hugo to make it easy to update and manage the blog content .

Sign up for an IBM Cloud account

To sign up for an IBM Cloud account, follow these steps:

- Open your web browser and go to the IBM Cloud website (https://cloud.ibm.com).

- Click on the "Sign up" or "Create an IBM Cloud account" button. This will typically be prominently displayed on the homepage.

- You will be taken to the sign-up page. Fill in the required information, including your email address, first name, last name, and password. You may also need to confirm your password.

- Optionally, you can enter a "Promo code" if you have one. Promo codes can provide discounts or special offers.

- After providing your information, click the "Next" or "Continue" button.

- On the next page, you may need to provide additional information, including your phone number and company details.

- Review the terms and conditions and privacy policy. If you agree, check the box to accept them.

- Click the "Create Account" or "Sign Up" button to complete the registration process.

- You may receive a verification email to confirm your email address. Follow the instructions in the email to verify your account.

- Once your email is verified, you should be able to log in to your newly created IBM Cloud account.

Create a new static web app and follow the prompts to set up the repository,build pipeline and deployment options.

Creating a new static web app with a repository, build pipeline, and deployment options typically involves using a combination of tools and services, such as GitHub, Azure Static Web Apps, or Netlify. Here's a general guide on how to do this:

Create a Repository:

- Start by creating a new repository on a platform like GitHub, GitLab, or Bitbucket. You can do this through the web interface of your chosen platform.

Add Your Web App Code:

- Push your static web app code to this repository. Make sure the code includes all the necessary HTML, CSS, JavaScript, and any other assets.

Choose a Hosting and Deployment Platform:

- Depending on your preference, you can choose a hosting and deployment platform.

Some popular options are:

GitHub Pages: If you're using GitHub, you can easily host your static site using GitHub Pages. Configure it in your repository's settings.

Azure Static Web Apps: If you prefer Azure, you can set up a new Static Web App in the Azure Portal and link it to your repository. It will automatically create a build and deployment pipeline for you.

Netlify: Netlify is another popular platform for hosting static sites. You can connect your repository and set up continuous deployment through Netlify.

Configure Build and Deployment:

- This step may vary depending on the platform you chose. But in most cases, you need to define a build and deployment process.

Here's a general outline:

- Set up a build pipeline to compile and optimize your code, if needed. This could involve running build scripts or using build tools like Gulp, Webpack, or Jekyll.
- Define the deployment settings. You may need to specify which branch triggers a deployment, whether it's the main branch or a specific one.
- Configure environment variables if your app requires them (e.g., API keys or configuration variables).

Follow the Platform's Prompts:

- For Azure Static Web Apps, after connecting your repository, you'll be prompted to set up your build configuration, API routes, and other settings through the Azure Portal. Follow the on-screen instructions.

Test and Deploy:

- After configuration, test the build and deployment pipeline by making changes to your code and pushing to the repository. You should see the changes automatically deployed based on your setup.

Custom Domain (Optional):

- If you have a custom domain for your web app, configure it through your hosting platform's settings.

- The specific steps and interface details may differ depending on the platform you choose, but this general guide should give you a good starting point. Be sure to consult the documentation of your chosen platform for more precise instructions.

Choosing Jekyll or Hugo

Ease of Use:

- Consider your level of technical expertise. Jekyll, being Ruby-based, may be more familiar to developers with Ruby experience, while Hugo is written in Go and might be more accessible to those with Go knowledge.

Performance:

- Both Jekyll and Hugo are known for their speed and efficiency. Hugo is often praised for its remarkable speed in building sites.

Community and Ecosystem:

- Check the community support, availability of themes, and plugins for the SSG you choose. A larger community often means more resources and solutions when issues arise.

Templates and Themes:

- Consider the availability of templates and themes for customization. Both Jekyll and Hugo have a good selection, but you might prefer one over the other based on your design requirements.

Content Management:

- Think about how easy it is to manage your content. Both Jekyll and Hugo use Markdown, which is straightforward for writing

blog posts. Jekyll has a more user-friendly administrative interface for non-technical users.

Build Process:

- Understand the build process. Hugo, with its incredible speed, offers an advantage for large blogs. Jekyll, while fast, may not be as speedy as Hugo in building large sites.

Hosting:

- Ensure that the SSG you choose is compatible with your preferred hosting platform. Both Jekyll and Hugo can be hosted on popular platforms like GitHub Pages, Netlify, and others.

Customization:

- Evaluate the degree of customization you require. Both Jekyll and Hugo allow extensive customization, but you might prefer one's theming system or configuration over the other.

Documentation:

- Check the quality and comprehensiveness of documentation. Good documentation can save you a lot of time troubleshooting issues.

Long-Term Goals:

- Consider your long-term goals. If you plan to grow your blog into a complex website with advanced features, one SSG might be better suited to handle that growth.

Personal Preference:

- Ultimately, it might come down to personal preference. You can try both Jekyll and Hugo on a small scale to see which one you feel more comfortable with.

In summary, both Jekyll and Hugo are excellent choices for managing and updating blog content. Your decision should be based on your familiarity with the technology, your specific requirements, and your comfort with the features and ecosystem of the SSG.