Building the travel blog by setting up the IBM Cloud Static Web App and deploying the website:

Setting up an IBM Cloud Static Web App and deploying your travel blog website is a multi-step process. Here’ s a high-level overview of the steps involved:

Prerequisites:

Make sure you have an IBM Cloud account.Have your website code ready in a repository (e.g., GitHub).

Create a New Static Web App:

Log in to your IBM Cloud account. In the IBM Cloud dashboard, navigate to the “ Create Resource” section.

Search for “ Static Web App” and select it Follow the setup wizard to configure your app, specifying the GitHub repository and other settings.

Configure Build and Deploy Settings:

In the Static Web App setup, configure your build settings. This typically involves specifying the build command and build output directory. For example, if you’ re using a static site generator like Jekyll, you might use `jekyll build` as the build command.

Add Environment Variables (Optional):

If your website requires environment variables (API keys, configurations), set them in the IBM Cloud environment for added security.

Review and Deploy:

Review the configuration settings to ensure they are correct.

Click the “ Create” or “ Deploy” button to start the deployment process.

Wait for Deployment:

The deployment process may take a few minutes to complete. You can monitor the progress within the IBM Cloud dashboard.

Access Your Website:

Once the deployment is successful, you will receive a URL where your website is hosted.

Custom Domain (Optional):

If you want to use a custom domain for your travel blog, you can configure it in the IBM Cloud settings or through your domain registrar.

Testing:

Test your website to ensure that everything is working as expected.

Maintenance:

Regularly update and manage your website content and configurations.

<u>Create a new static web app and follow the prompts to set up the repository, build pipeline and deployment options:</u>

creating a new Static Web App with the assumption that you’ re using the IBM Cloud platform.

Search for Static Web App:
In the resource catalog, search for “ Static Web App” and select it.

Configure Your Web App:
Follow the setup wizard to configure your web app. You’ ll need to provide information such as the app name, region, and resource group.
Connect Your Repository:
In the setup process, you will be prompted to connect your GitHub repository or other supported source control providers.
Configure Build Settings:
Specify your build command and build output directory. This will depend on your website’ s build process. Common build commands include “ npm run build” for JavaScript-based projects or “ jekyll build” for Jekyll sites.
Add Environment Variables:
If your website requires environment variables (e.g., API keys), you can set them here.
Deployment Options:
Choose deployment options such as branch or tag to deploy from and how often to trigger deployments. You can set up automatic deployments whenever changes are pushed to a specific branch.
Review and Deploy:
Review the configuration settings to ensure they are correct.
Create or Deploy:
Click the “ Create” or “ Deploy” button to initiate the deployment process. This will set up your Static Web App on IBM Cloud.
Wait for Deployment:
The deployment process may take a few minutes to complete. You can monitor the progress within the IBM Cloud dashboard.
Access Your Website:
Once the deployment is successful, you will receive a URL where your website is hosted
Testing:

Test your website to ensure that everything is working as expected.

Choose a static site generator like Jekyll or Hugo to make it easy to update and manage the blog content:

Choosing a static site generator like Jekyll or Hugo is a great idea for making it easier to update and manage your travel blog content. These generators allow you to convert your HTML content into template files, making content management more efficient.

Jekyll:

Jekyll is a widely used static site generator that’ s written in Ruby.

It uses Markdown for content, making it easy to write and manage blog posts Jekyll uses Liquid as its templating language.Configuration and theming are flexible, allowing you to customize your blog’ s appearance.GitHub Pages, which supports Jekyll, can be a convenient hosting option.

Hugo:

Hugo is written in Go and is known for its speed and performance.

It uses Markdown for content like Jekyll but also supports other formats.

Hugo uses Go’ s text/template and html/template libraries for templating.

Hugo offers a wide range of themes and is highly customizable.

It’ s a great choice for those who prefer a static site generator with fast build times.

Source code:

```
``html

<!DOCTYPE html>

<html>

<head>
```

```
  <title>My Travel Blog</title>
  <link rel=” stylesheet” type=” text/css” href=” style.css” >
</head>
<body>
  <header>
    <h1>My Travel Blog</h1>
    <nav>
      <ul>
        <li><a href=” #” >Home</a></li>
        <li><a href=” #” >Destinations</a></li>
        <li><a href=” #” >Blog</a></li>
        <li><a href=” #” >About</a></li>
        <li><a href=” #” >Contact</a></li>
      </ul>
    </nav>
  </header>

  <section class=” main-content” >
    <article>
      <h2>Exploring the Beaches of Bali</h2>
      <img src=” bali.jpg” alt=” Bali Beach” >
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. … </p>
      <a href=” #” >Read More</a>
    </article>
    <article>
      <h2>Hiking in the Swiss Alps</h2>
      <img src=”swiss.jpg” alt=”Swiss Alps”>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. … </p>
      <a href=”#”>Read More</a>
    </article>
  </section>

  <aside>
    <h2>Popular Destinations</h2>
    <ul>
      <li><a href=”#”>Paris, France</a></li>
      <li><a href=”#”>Tokyo, Japan</a></li>
      <li><a href=”#”>Machu Picchu, Peru</a></li>
      <li><a href=”#”>Santorini, Greece</a></li>
    </ul>
  </aside>

  <footer>
    <p>&copy; 2023 My Travel Blog</p>
  </footer>
</body>
</html>
```

HTML Templates:

Create HTML templates for the cart and checkout pages, e.g., `cart.html` and `checkout.html`.