# *Personal Blog on IBM Cloud Static Web App*

**Team Members**

Furqan Ahmed Mohammed – furqan076@gmail.com

M. Jalaludeen Zubair – jalaludeenzubair15@gmail.com

Md Sadan Fuzail S – fuzailahmed20962@gmail.com

S. Bharath – bharathshanmugam11@gmail.com

P. Aathi Siva Ganesh – p.aathisivaganesh@gmail.com

M. Arssam Basha – arssambasha82@gmail.com

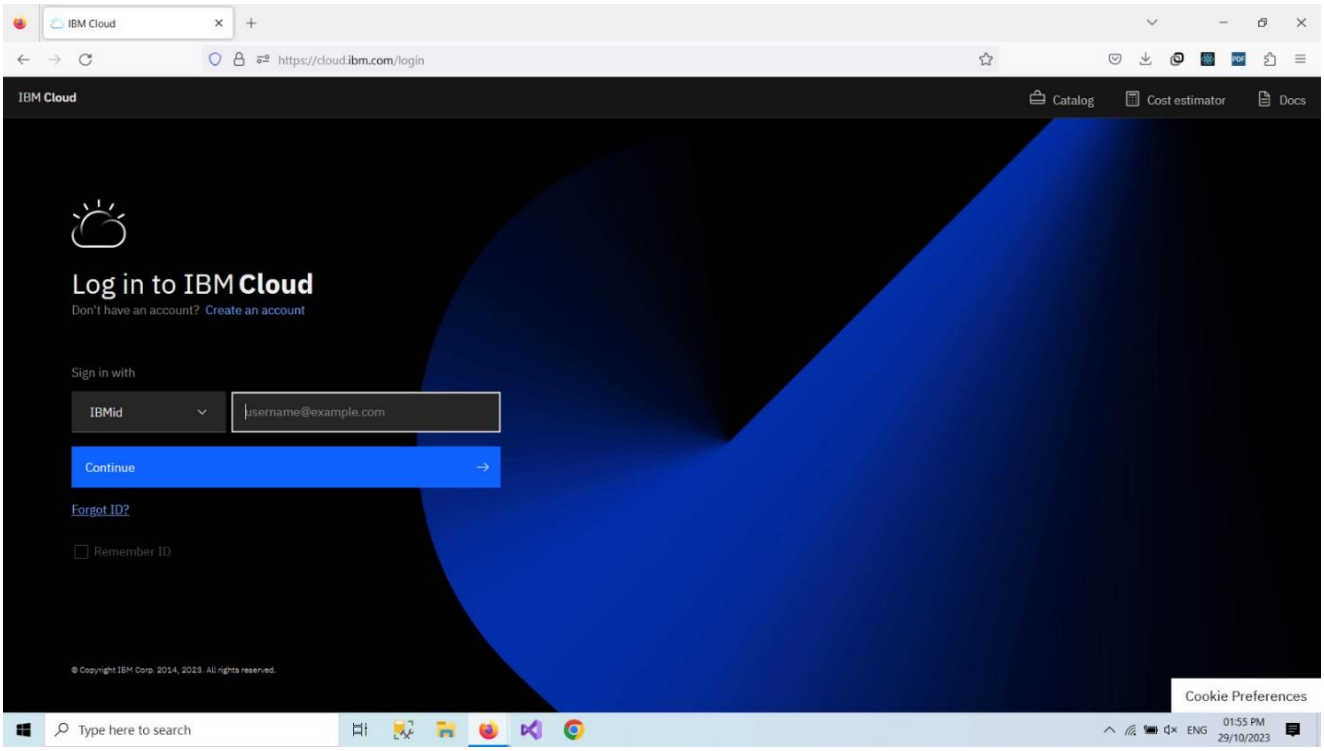
## *Phase 4: Development Part 2*

### Objective:

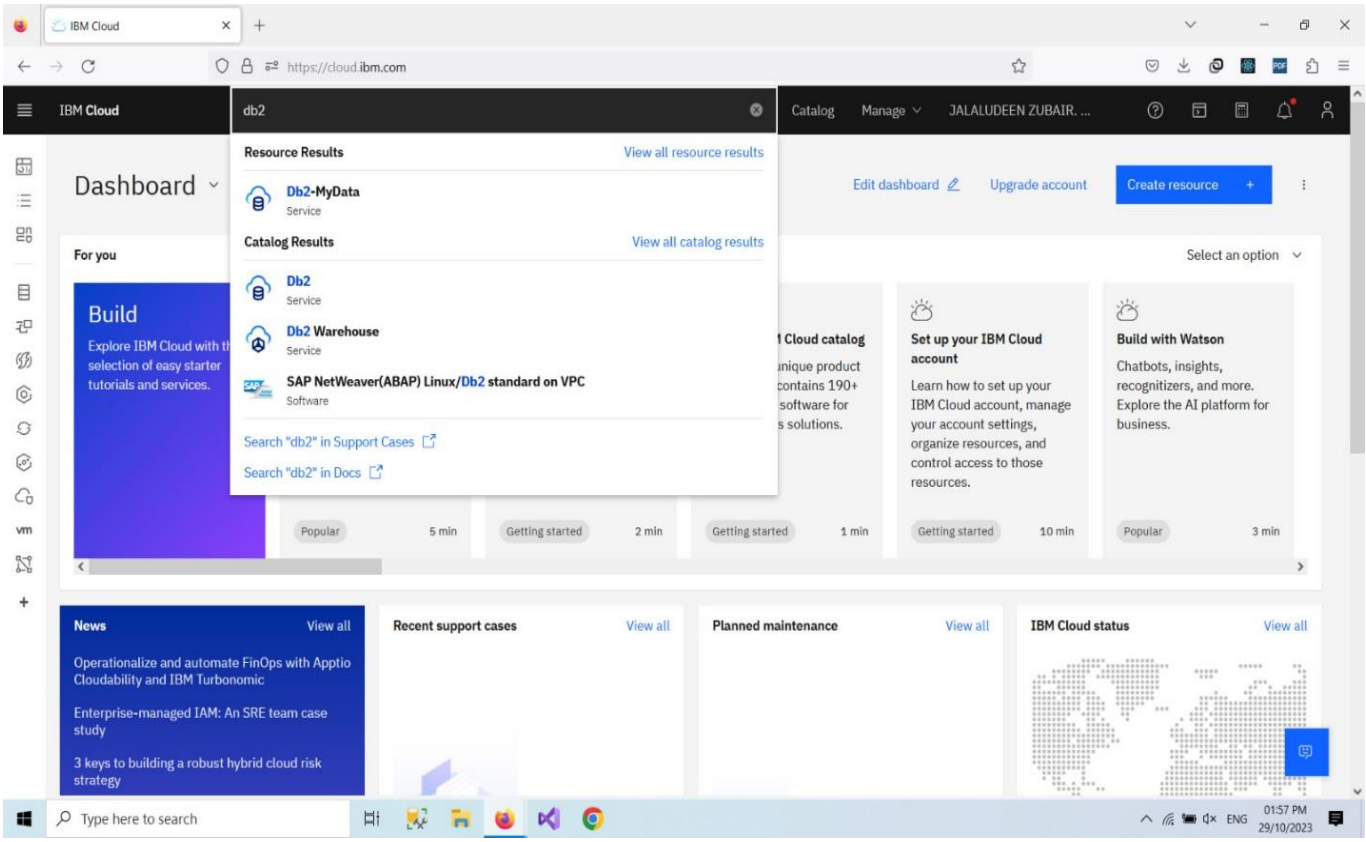
The primary objective of the personal blog, hosted on the IBM Cloud as a static web app, is to share the passion for travel and adventure while demonstrating the capabilities of IBM Cloud for hosting static websites.

### Interaction Features:

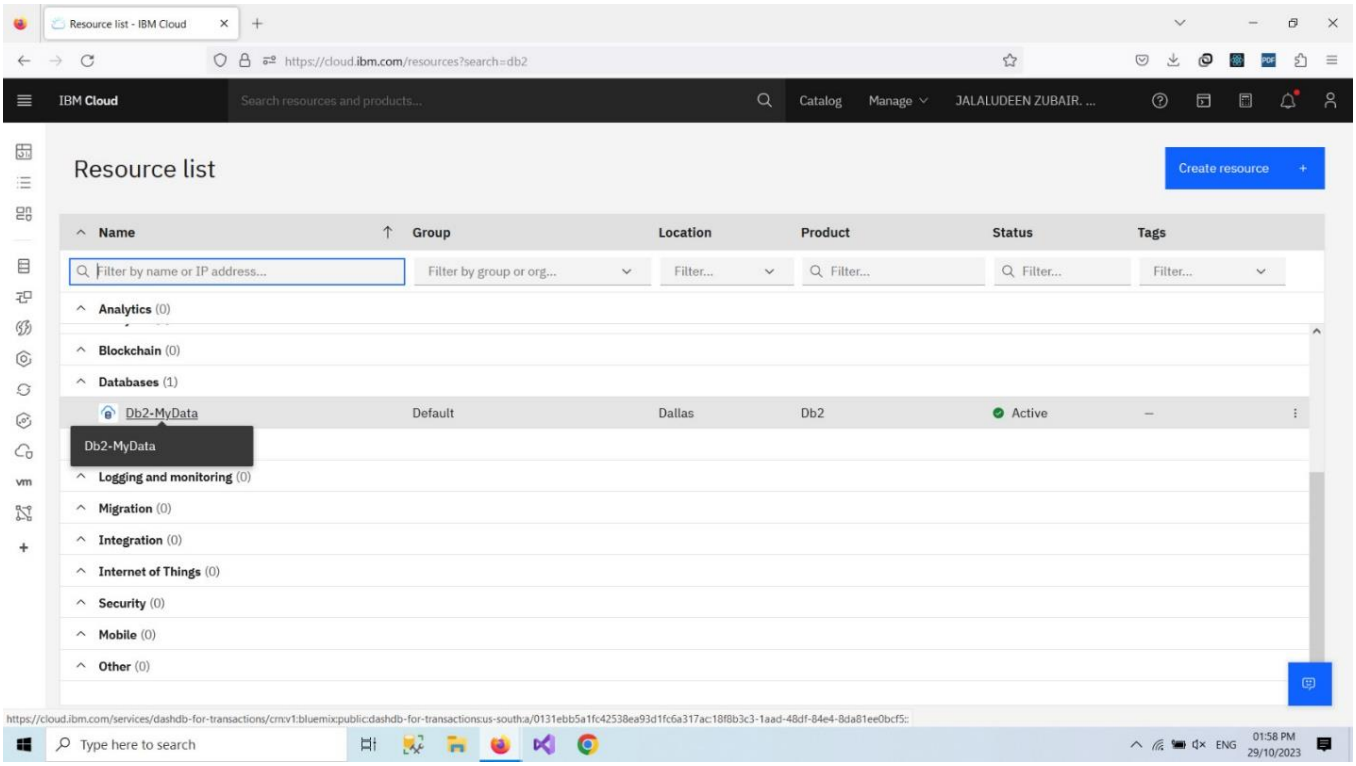
- Build a website to host a personal blog on the IBM Cloud using HTML, CSS, Flask, IBM db2/SQLlite.
- Upload the document containing all the details regarding the deployment of the website.
- Deploy the code using docker image, container registry and Kubernetes.



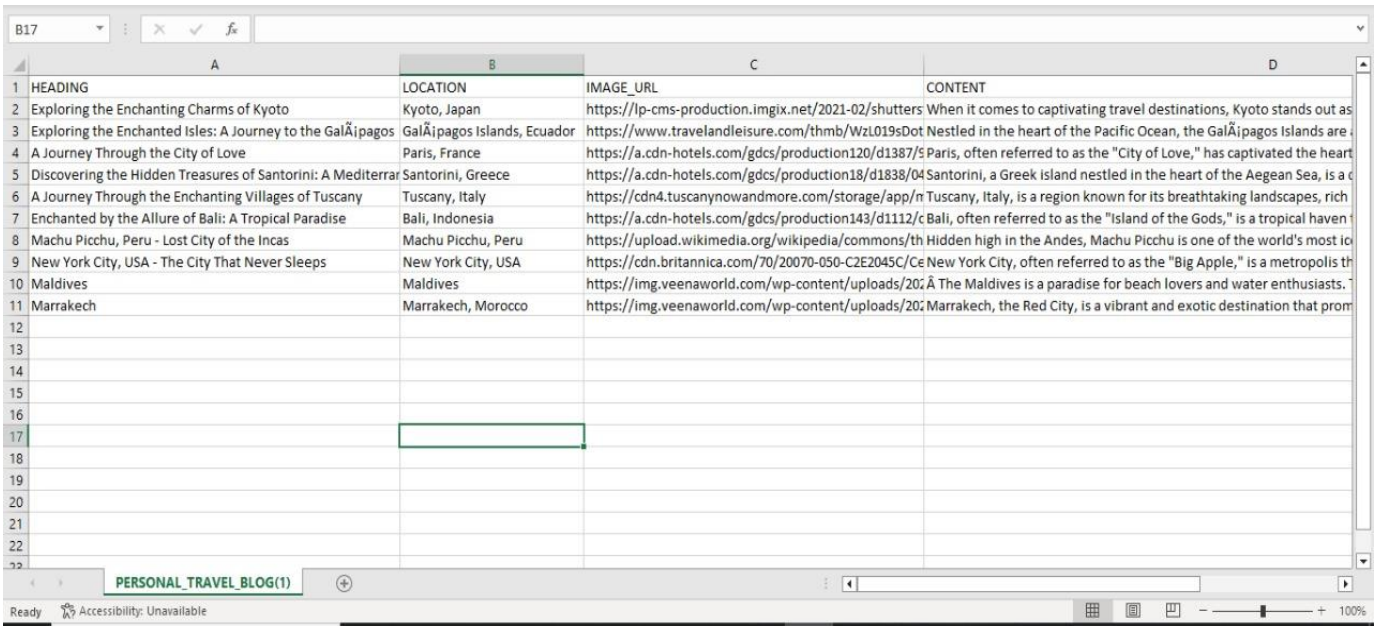

- First, you need to have an IBMid, which is a single account that you can use to access IBM products and services. If you don't have one, you can create one by clicking on the "Create an IBMid" link below the login form.
- Next, you need to enter your IBMid and password in the respective fields of the login form. Make sure you type them correctly and avoid any typos or errors.
- Then, you need to click on the "Continue" button to submit the login form and access the IBM Cloud site. If you have entered your credentials correctly, you will be redirected to the IBM Cloud dashboard, where you can view and manage your cloud resources and services.




- First, the user has clicked on the “Build” tab, which is one of the four tabs on the top of the page. The other tabs are “Manage”, “Catalog”, and “Support”.
- Next, the user is looking at the “Build with Watson” section, which is one of the six sections on the left side of the page. The other sections are “Cloud Foundry apps”, “Kubernetes Service”, “Functions”, “Cloud Services”, and “DevOps”.
- Then, the user can see various options to build applications using Watson, which is IBM’s artificial intelligence platform. The options are “Watson Assistant”, “Watson Discovery”, “Watson Knowledge Studio”, “Watson Natural Language Understanding”, “Watson Speech to Text”, and “Watson Text to Speech”.
- Finally, the user can click on any of these options to create a new instance of the service or view their existing instances.

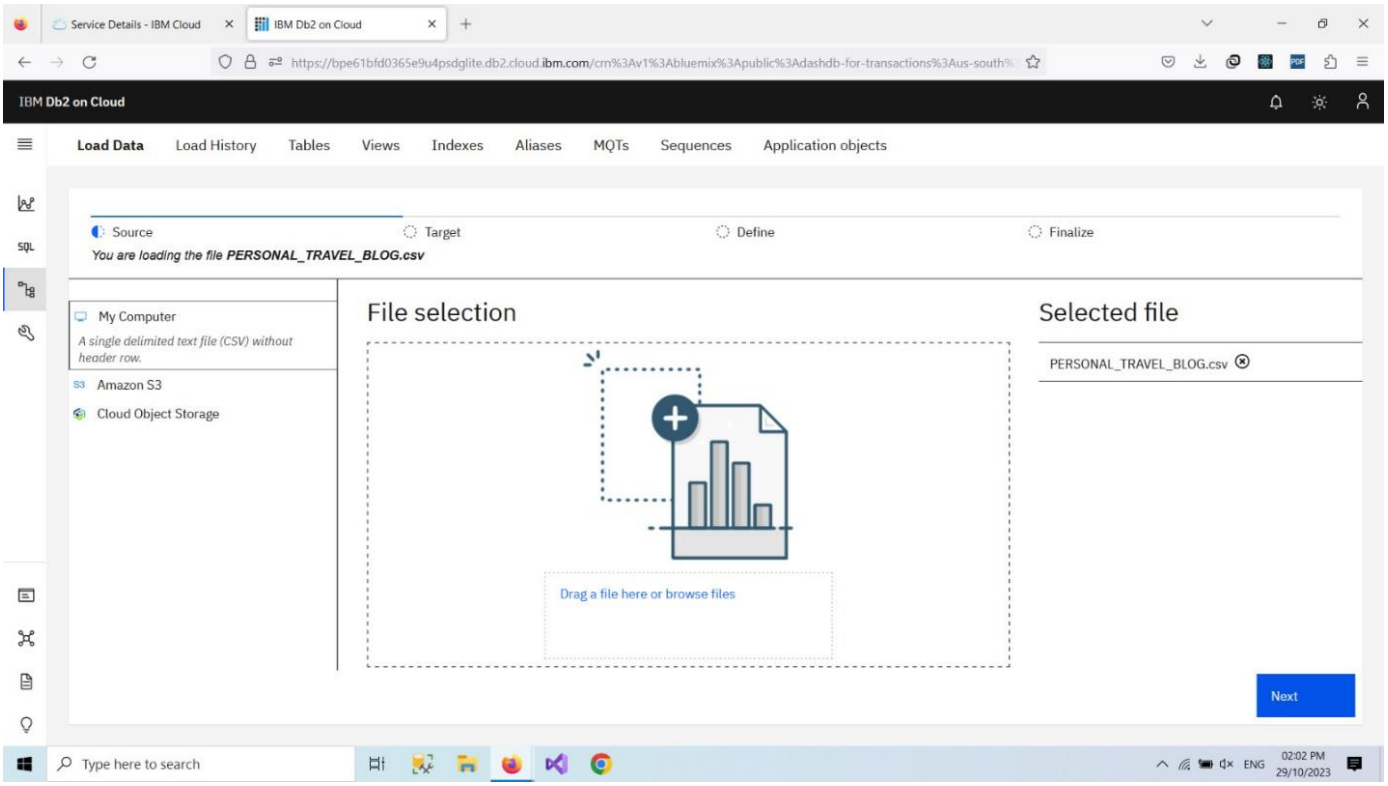



- First, the user has accessed the webpage by clicking on the “Resource list” option in the left navigation bar of the IBM Cloud website. This option shows all the resources that the user has created or used on IBM Cloud.
- Next, the user has filtered the list by the “Name” column, which is sorted alphabetically. This column shows the name of each resource, which can be customized by the user. The user can also filter the list by other columns such as “Group”, “Location”, “Product”, “Status”, and “Tags”.
- Then, the user has clicked on the “Databases” row, which is highlighted in blue. This row represents a resource that is a database service on IBM Cloud. The user can see more details about this resource by clicking on the row, such as its type, plan, region, and credentials.
- Finally, the user has also clicked on the “Location” filter, which is highlighted in blue. This filter shows the location of each resource, which can be different from the user’s location. The user can use this filter to narrow down the list by selecting or deselecting specific locations.



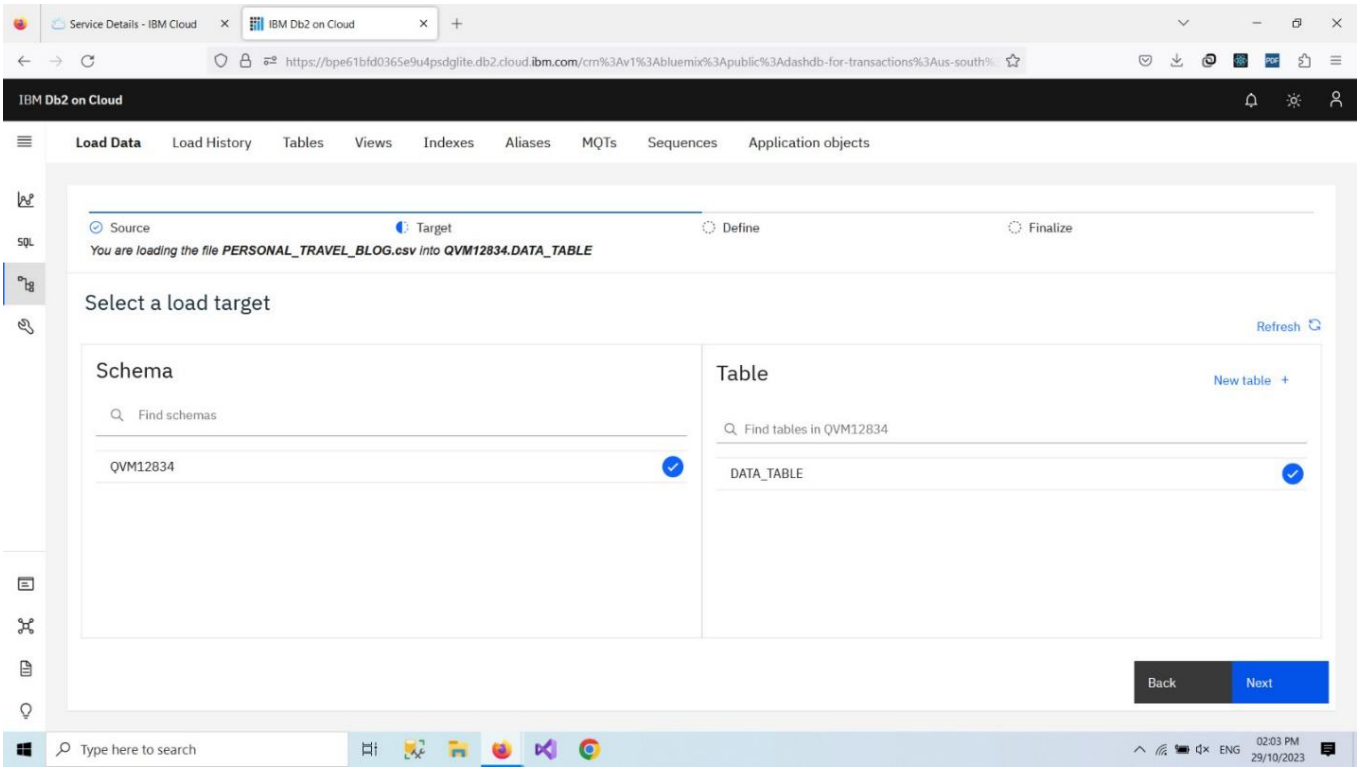
| | A | B | C | D |
|---|---|---|---|---|
| 1 | HEADING | LOCATION | IMAGE_URL | CONTENT |
| 2 | Exploring the Enchanting Charms of Kyoto | Kyoto, Japan | https://lp-cms-production.imgix.net/2021-02/shutters | When it comes to captivating travel destinations, Kyoto stands out as |
| 3 | Exploring the Enchanted Isles: A Journey to the GalÃ¡pagos | GalÃ¡pagos Islands, Ecuador | https://www.travelandleisure.com/thmb/WzL019sDot | Nestled in the heart of the Pacific Ocean, the GalÃ¡pagos Islands are |
| 4 | A Journey Through the City of Love | Paris, France | https://a.cdn-hotels.com/gdcs/production120/d1387/5 | Paris, often referred to as the "City of Love," has captivated the heart |
| 5 | Discovering the Hidden Treasures of Santorini: A Mediterrar | Santorini, Greece | https://a.cdn-hotels.com/gdcs/production18/d1838/04 | Santorini, a Greek island nestled in the heart of the Aegean Sea, is a |
| 6 | A Journey Through the Enchanting Villages of Tuscany | Tuscany, Italy | https://cdn4.tuscanynowandmore.com/storage/app/n | Tuscany, Italy, is a region known for its breathtaking landscapes, rich |
| 7 | Enchanted by the Allure of Bali: A Tropical Paradise | Bali, Indonesia | https://a.cdn-hotels.com/gdcs/production143/d1112/c | Bali, often referred to as the "Island of the Gods," is a tropical haven |
| 8 | Machu Picchu, Peru - Lost City of the Incas | Machu Picchu, Peru | https://upload.wikimedia.org/wikipedia/commons/th | Hidden high in the Andes, Machu Picchu is one of the world's most ic |
| 9 | New York City, USA - The City That Never Sleeps | New York City, USA | https://cdn.britannica.com/70/20070-050-C2E2045C/Ce | New York City, often referred to as the "Big Apple," is a metropolis th |
| 10 | Maldives | Maldives | https://img.veenaworld.com/wp-content/uploads/20 | Â The Maldives is a paradise for beach lovers and water enthusiasts. |
| 11 | Marrakech | Marrakech, Morocco | https://img.veenaworld.com/wp-content/uploads/20 | Marrakech, the Red City, is a vibrant and exotic destination that prom |

- First, you need to have spreadsheet software, such as Microsoft Excel or Google Sheets, installed on your computer or accessible online. You can use this software to create, edit, and save spreadsheets.
- Next, you need to enter the data into the spreadsheet, including the headers and the content. The headers are the labels for each column, such as “HEADING”, “LOCATION”, “IMAGE URL”, etc. The content is the information for each row, such as the name, location, image URL, etc. of a travel destination or activity. You can type the data manually or copy and paste it from another source.
- Then, you need to format the spreadsheet to make it look neat and organized. You can use various formatting options, such as font size, color, alignment, borders, etc. to change the appearance of the data. You can also adjust the width and height of the rows and columns to fit the data. You can also apply filters and sorts to the data to display it in a specific order or criteria.
- Finally, you need to take a screenshot of the spreadsheet and save it as an image file. You can use a keyboard shortcut or a tool to capture the screen and save it in a folder on your computer. You can also edit the image file if needed.

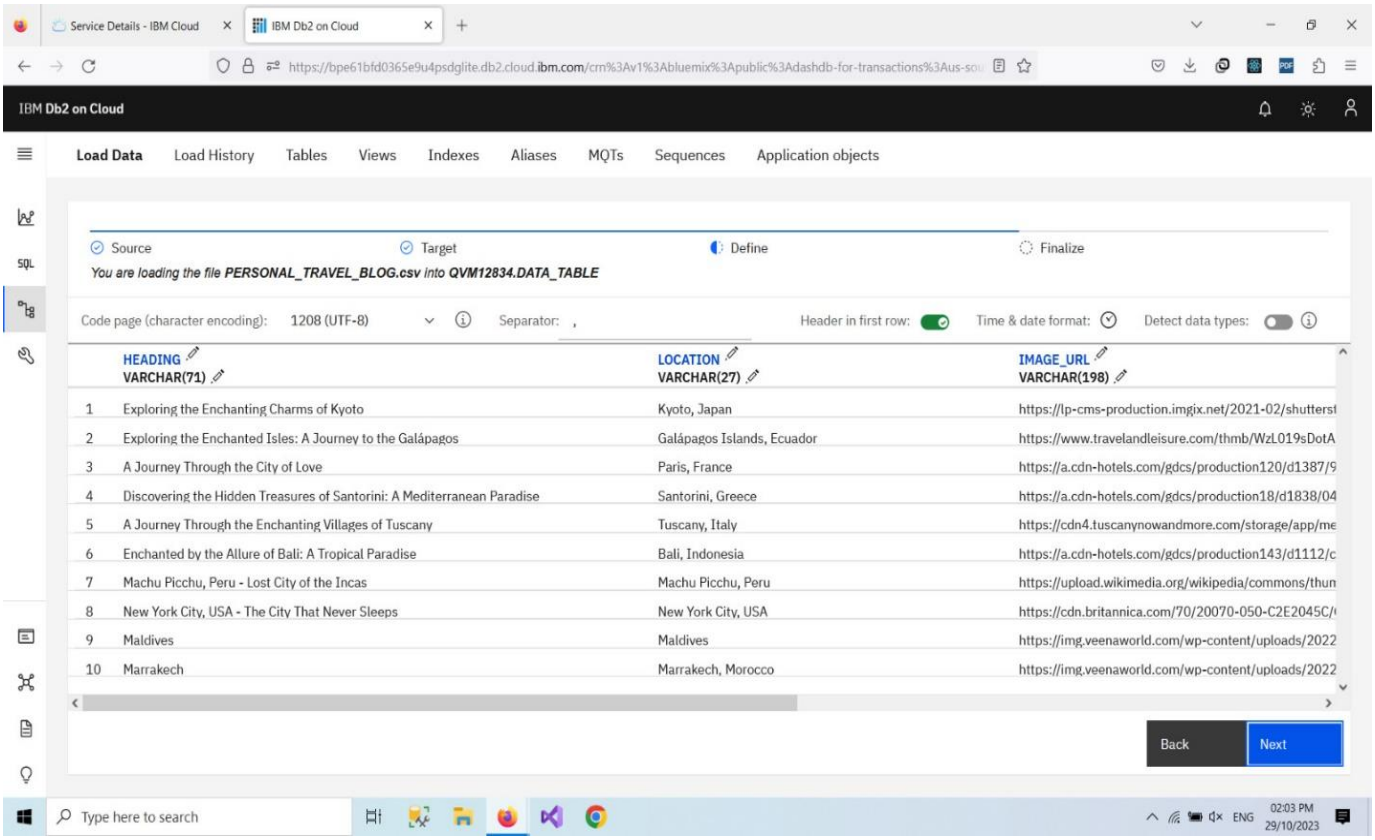



The image is a screenshot of a computer screen showing a webpage of IBM DB2 on Cloud, which is a cloud-based database service that allows you to store and analyze data.

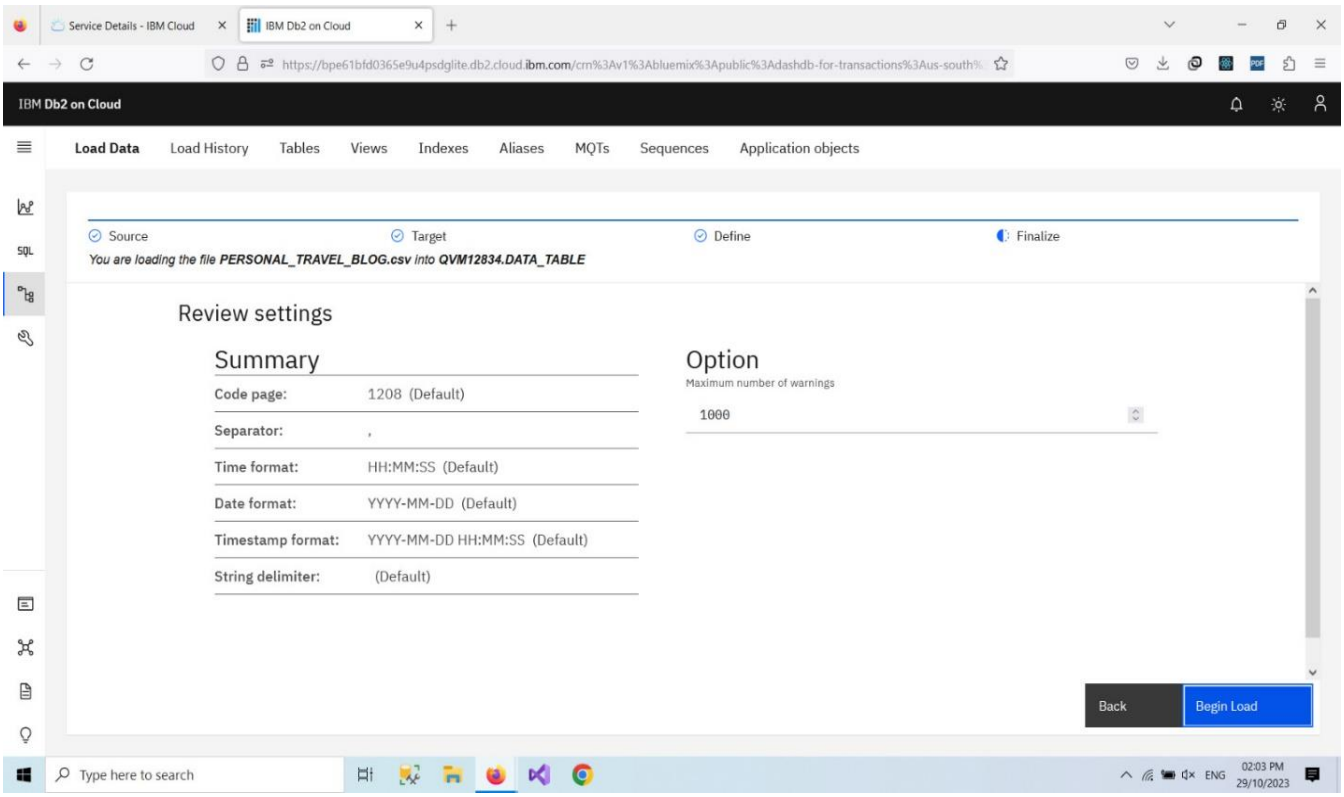
- First, you need to create an instance of IBM DB2 on Cloud, which is a specific configuration of the database service that suits your needs. You can do this by clicking on the “Create” button in the top left corner of the webpage and following the instructions.
- Then, you need to upload a file into your IBM DB2 on Cloud instance, which is a data source that you want to store and analyze in the database. You can do this by clicking on the “Load Data” option in the left navigation bar of the webpage and selecting the “Personal” tab.
- Finally, you need to drag and drop a file into the file upload interface, which is a large gray box with a blue dashed border. The file should be in CSV format, which is a common format for storing tabular data. The file upload interface is labeled with the name of the file, which is “PERSONAL_TRAVEL_BLOG.csv”. You can also see the selected file in a blue box on the right side of the screen. Once you have uploaded the file, you need to click on the “Next” button to proceed with the data loading process.




- First, you need to access the “Application objects” section of IBM Db2 Cloud, which is where you can create and manage tables and other objects in your database. You can do this by clicking on the “Application objects” option in the left navigation bar of the IBM Db2 Cloud webpage.
- Then, you need to select a table in IBM Db2 Cloud, which is a collection of data organized in rows and columns. You can do this by following these steps:
    - Click on the “Tables” tab in the “Application objects” section, which will show you a list of all the tables in your database.
    - Click on the “New table” button, which will open a dialog box where you can create a new table or select an existing one.
    - Select the “PERSONAL_TRAVEL_BLOG” table in the “Source” column, which is the table that contains the data that you want to use for your new table. This table was uploaded from a CSV file in a previous step.
    - Select the “OWNPERSONAL_TABLE” table in the “Target” column, which is the name of your new table that you want to create from the source table. You can also change the name if you want.
    - Click on the “Next” button to proceed to the next step, where you can define the columns and data types for your new table.

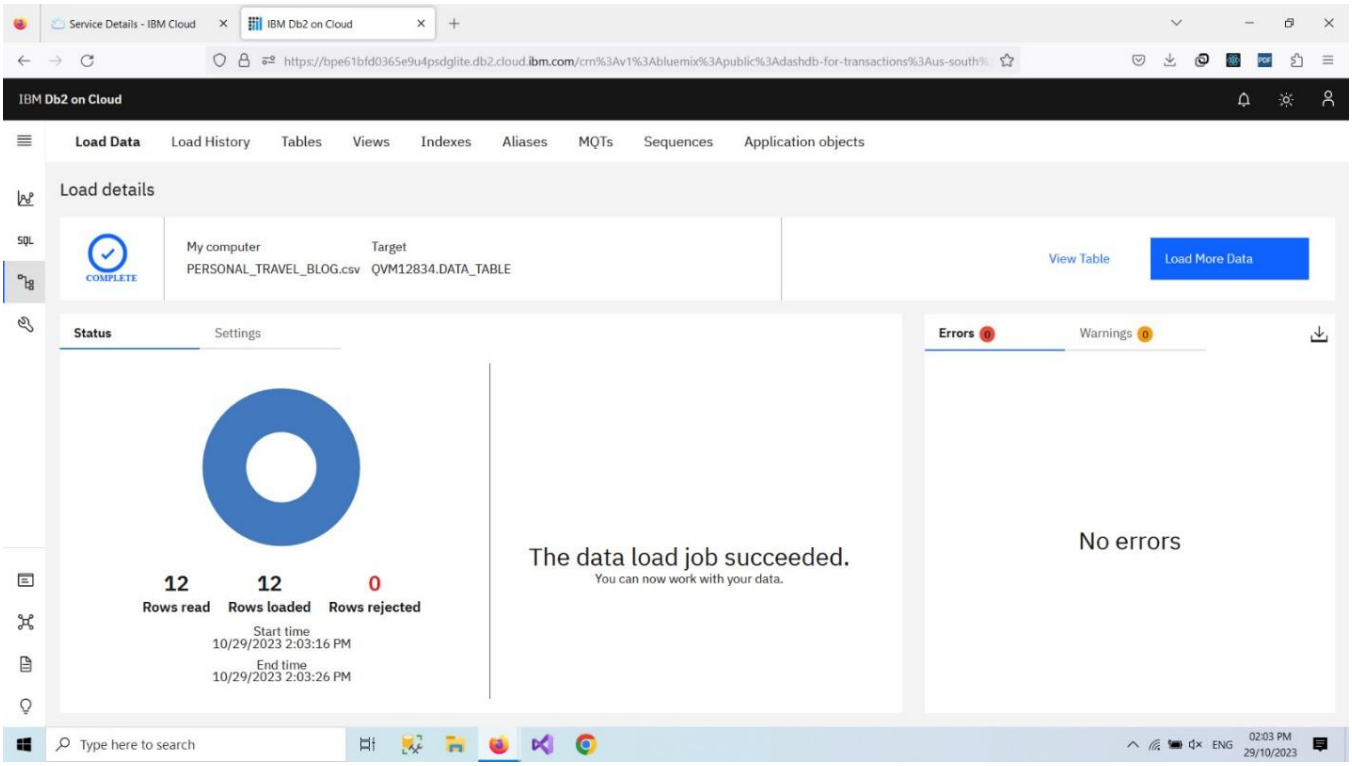



- First, you need to load data into your IBM DB2 on Cloud instance, which is a data source that you want to store and analyze in the database. You can do this by clicking on the “Load Data” tab on the top navigation bar of the IBM DB2 on Cloud webpage and selecting the “Personal” or “Enterprise” option depending on your data type and size.
- Next, you need to view the data that you have loaded into your IBM DB2 on Cloud instance, which is a table of data with columns for “Data Source”, “Target”, “Header row”, “File date & time”, and “Object description”. You can do this by staying on the “Load Data” page and looking at the table of data. You can see 9 rows of data that you have loaded from different files. You can also navigate to the next page using the “Next” button on the bottom right corner of the table.



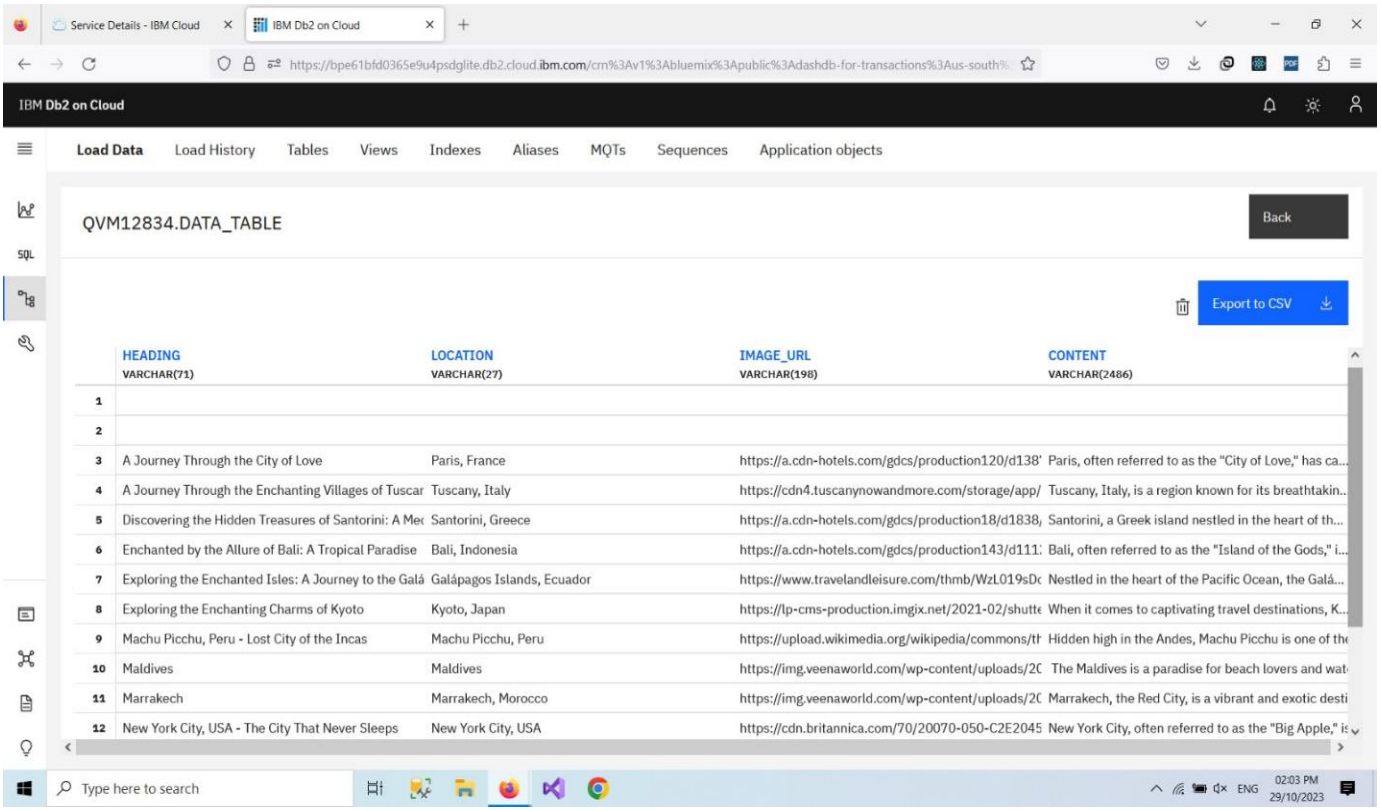

Review the settings for the table that you are loading data into, which is a collection of data organized in rows and columns. You can do this by following these steps:

- Click on the “Review the settings” tab, which is the third and final tab in the data loading process.
- Check the settings for your table, such as its name, schema, columns, data types, and primary key. You can see these settings in table format on the left side of the screen.
- If you want to change any of these settings, you can click on the “Edit” button below the table and make your changes.
- Once you are satisfied with your settings, you can click on the “Begin Load” button on the bottom right corner of the screen to start loading the data into your table.




View the details of the data load job that you have initiated, which is a process that transfers the data from your file into your table. You can do this by following these steps:

- Click on the “Load” tab, which is the second tab in the top navigation bar of the IBM DB2 on Cloud webpage. This tab shows you a list of all the data load jobs that you have started or completed.
- Click on the “Load Data” tab, which is the first sub-tab under the “Load” tab. This tab shows you a list of all the data load jobs that are related to loading data from files into tables.
- Select the “My personal table” option, which is one of the options in the left navigation bar of the “Load Data” tab. This option shows you a list of all the data load jobs that are related to loading data into your personal tables.
- Click on the “View details” button, which is a gray button next to each data load job in the list. This button opens a new page where you can see more information about the data load job, such as its status, settings, and results.
- You are now viewing the details of the data load job that you have initiated. You can see a blue header with the IBM DB2 on Cloud logo and the text “IBM DB2 on Cloud”. You can also see a white background with a gray “Load details” card. The card has a blue circle with the number “12” in it, which indicates that this is your 12th data load job. The card also has a gray “Settings” button and a blue “Refresh” button on its top right corner. The card also has various sections that show you different aspects of your data load job, such as its source file, target table, column mapping, header row, file date and time, object description, status history, and error messages.




The image is a screenshot of a data table in a web browser, which is a type of document that displays data in rows and columns. Here are the steps to follow:

- First, you need to have a web browser, which is a software application that allows you to access and view webpages on the internet. You can use any web browser that you prefer, such as Google Chrome, Mozilla Firefox, Microsoft Edge, etc.
- Next, you need to navigate to the webpage that contains the data table, which is a webpage with a white background and a header with the title “LM1384_DATA_TABLE”. You can do this by typing the URL of the webpage in the address bar of your web browser and pressing enter. The URL is [this].
- Then, you need to view the data table, which is a table of data with 6 columns: HEADING, LOCATION, TABLE, IMAGE_URL, CONTENT, and WORDCOUNT. You can do this by scrolling down the webpage and looking at the table. You can see 10 rows of data that are related to travel destinations and activities. Each row has different data in each column.
- Finally, you need to export the data table to CSV, which is a common format for storing tabular data. You can do this by clicking on the “Export to CSV” button on the top right corner of the table. This button will download the data table as a CSV file on your computer. You can also go back to the previous webpage by clicking on the “Back” button on the top left corner of the webpage.

```
- _travel_blog_/
 - app/
  - app.py   # Your main Python application file
 - static/
  - css/
   - style.css   # Your CSS file
   - index.css
  - js/
   - main.js   # Your JS file
 - templates/
  - index.html    # One of your HTML files
  - post.html  # The other HTML file
 - Dockerfile
 - requirements.txt
```

1. **_travel_blog_:** This is the root directory of your Flask application.

    - **app:** This directory contains the core application code and resources.
    - **Templates:** This subdirectory is used to store HTML templates that define the structure and layout of your web pages.
    - **index.html:** This HTML template represents the main page of your application, where you list or display content such as blog posts.
    - **post.html:** This HTML template is used to display individual blog posts or detailed content.
    - **static:** This subdirectory is for storing static assets such as CSS and other frontend resources.
    - **style.css:** This CSS file is used to style the HTML templates for the entire application.
    - **Index.css:** This CSS file, if it's intended to be used, would likely provide additional styling specific to the index page (index.html) or serve a different purpose than style.css.

2. Dockerfile: This file provides instructions for building a Docker image of your Flask application. It typically specifies the base image, sets up the working directory, copies application files, installs dependencies, and defines the command to run the Flask application.

3. requirements.txt: This file lists the Python packages and their versions that your application depends on. It's used to ensure that the required packages are installed when setting up your application's environment.

4. app.py: This is the primary Flask application file. It contains the Python code for defining your application, including routes, views, and any backend logic necessary for your web application. You would typically create and configure your Flask app here, set up routes for serving HTML templates and handling user requests, and define the application's behavior.

index.html appears to be the main page, where it lists multiple blog posts. It uses Flask templating to loop through the posts retrieved from the database and display their information. The page includes a header, images, post titles, and post locations.

## index.html

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>My Blog</title>

<link rel="stylesheet" href="/app/static/index.css" />

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700&display=swap"
rel="stylesheet">

</head>

<body>

<header>PERSONAL BLOG WEBSITE</header>

<div class="all-posts-container">

{% for post in posts %}

<div class="post-container">

<a href="{{ url_for('view_post', post_id=post.id) }}" class="post-preview">

<div class="picture-space">

<img class="picture"

src="{{post.image_url }}"
```

```
alt="{{post.heading }} image"

width="500px"

/>

</div>

<div class="post-info">

<p class="post-title">{{ post.heading }}</p>

<p class="post-location">{{ post.location }}</p>

</div>

</a>

</div>

{% endfor %}

</div>

</body>

</html>
```

post.html seems to be a template for displaying individual blog posts. It includes social media sharing buttons, post content, and additional styling. The template takes a single post object as an argument and displays its content, such as the title, location, image, and actual post content.

## post.html

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>My Blog</title>

<link rel="stylesheet" href="/app/static/style.css" />

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700&display=swap"
rel="stylesheet">

</head>

<body>

<div class="share-btn-container">

<a href="#" class="facebook-btn">

<i class="fab fa-facebook"></i>

</a>

<a href="#" class="twitter-btn">

<i class="fab fa-x-twitter"></i>

</a>
```

```
<a href="#" class="linkedin-btn">
<i class="fab fa-linkedin"></i>
</a>
<a href="#" class="pinterest-btn">
<i class="fab fa-pinterest"></i>
</a>
<a href="#" class="whatsapp-btn">
<i class="fab fa-whatsapp"></i>
</a>
</div>
<div class="content">
<h1 id="title">{{ post.heading }}</h1>
<p>
<a href="#" class="maps-btn">
<i id="location">{{ post.location }}</i>
<i class="fa-solid fa-map-location-dot"></i>
</a>
</p>
<div class="main-content-container">
<img class="img" src="{{ post.image_url }}" alt="Image" />
<p>{{ post.content }}</p>
</div>
</div>
<script src="main.js"></script>
```

```
</body>

</html>
```

The application uses CSS to style the HTML templates. It references two CSS files, index.css and style.css. CSS files define the visual presentation of the web pages, including fonts, colors, layout, and other styling elements. Proper CSS styling enhances the overall look and user experience of the website.

## index.css

```
.post-info {

font-family: Roboto, Arial;

font-size: 16px;

}

.post-title {

font-weight: 500;

}

.post-location {

color: rgb(92, 92, 92);

}

.post-container {

padding: 15px;

background-color: rgb(255, 255, 255);

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

transition: transform .2s;

}

.post-container:hover {

transform: scale(1.05);

}

.all-posts-container {
```

```
display: grid;
grid-template-columns: 1fr 1fr 1fr;
column-gap: 16px;
row-gap: 40px;
}
a {
text-decoration: inherit;
color: inherit;
}
.picture {
border: 2px solid rgb(52, 52, 52);
width: 100%;
}
@media (max-width: 450px) {
.all-posts-container {
grid-template-columns: 1fr;
}
}
@media (min-width: 451px) and (max-width: 750px) {
.all-posts-container {
grid-template-columns: 1fr 1fr;
}
}
@media (min-width: 751px) and (max-width: 1249px) {
```

```
.all-posts-container {
grid-template-columns: 1fr 1fr 1fr;
}
}
@media (min-width: 1250px) {
.all-posts-container {
grid-template-columns: 1fr 1fr 1fr 1fr;
}
}
header {
text-align: center;
font-weight: 700;
font-family: Roboto, Arial;
font-size: 60px;
background-color: rgb(51, 51, 51);
color: rgb(255, 255, 255);
padding: 20px;
margin-bottom: 20px;
}
body {
background-color: rgb(245, 245, 245);
padding-left: 10px;
padding-right: 10px;
}
```

style.css

```
/* Content */

.content {

padding: 8px 90px;

font-family: "Roboto", sans-serif;

}

.content p {

line-height: 1.9;

}

.content img {

max-height: 500px;

}

/* Share Buttons */

.share-btn-container {

background: #fff;

display: flex;

flex-direction: column;

padding: 16px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);

position: fixed;

top: 50%;

transform: translateY(-50%);

}

.share-btn-container a i {
```

```
font-size: 32px;

}

.share-btn-container a {

margin: 12px 0;

transition: 500ms;

}

.share-btn-container a:hover, .maps-btn :hover{

transform: scale(1.2);

}

.share-btn-container .fa-facebook {

color: #3b5998;

}

.share-btn-container .fa-x-twitter {

color: #000000

}

.share-btn-container .fa-linkedin {

color: #0077b5;

}

.share-btn-container .fa-pinterest {

color: #bd081c;

}

.share-btn-container .fa-whatsapp {

color: #25d366;

}
```

```
.maps-btn .fa-map-location-dot {
color: #cdca15;
transition: 500ms;
}
.maps-btn {
text-decoration: none;
color: #1c95a2;
}
.img {
width: 50vw;
max-width: 100%;
border: 2px solid rgba(0, 0, 0);
}
/* Media Queries */
@media (max-width: 550px) {
.content {
padding: 8px 32px;
}
.share-btn-container {
transform: unset;
top: unset;
left: 0;
bottom: 0;
width: 100%;
```

```
flex-direction: row;
box-shadow: 4px 0 8px rgba(0, 0, 0, 0.3);
padding: 16px 0;
justify-content: center;
}
.share-btn-container a {
margin: 0 32px;
}
.img {
width: 100vw;
max-width: 100%;
border: 2px solid rgba(0, 0, 0);
}
}
/* Comment Section */
.container{
display: flex;
justify-content: space-between;
padding: 10px;
margin: 10px 0;
}
.comment-container {
width: 80%;
margin: 0 auto;
```

```
font-family: Arial, sans-serif;
}
.comment-container h2 {
text-align: center;
}
#commentInput{
flex: 0.99;
padding:10px;
}
.comment {
display: flex;
justify-content: space-between;
background-color: #f2f2f2;
padding: 10px;
margin: 10px 0;
border: 1px solid #ddd;
border-radius: 5px;
}
.comment-actions {
display: flex;
}
.delete-button {
background-color: #ff5757;
color: white;
```

```
border: none;

padding: 5px 10px;

margin-left: 5px;

cursor: pointer;

}

.material-symbols-outlined:hover{

cursor:pointer;

}

.send{

font-size:30px;

}
```

The provided JavaScript code appears to serve two main functions: sharing and mapping, and a comment section for a web page. Let's break down the code step by step and explain its functionality.

Share and Maps Section:

This part of the code is responsible for adding social media sharing functionality to a web page and generating a link for mapping a location. It starts by selecting various elements from the HTML document.

1. facebookBtn, twitterBtn, pinterestBtn, linkedinBtn, whatsappBtn, and mapsBtn are variables that reference buttons or links in the HTML.

The init() function initializes these buttons with appropriate links:

- It collects the URL of the current web page (postUrl).

- It extracts the title of the page (postTitle).

- It captures the source URL of an image (postImg).

- It identifies the location content on the page (location).

After gathering this information, the code constructs links for sharing on different social media platforms. For example, facebookBtn is linked to Facebook sharing, and twitterBtn is associated with Twitter sharing. These links include relevant data such as the URL, title, and image for a more personalized share experience.

main.js

```
// Share and Maps section
const facebookBtn = document.querySelector(".facebook-btn");
const twitterBtn = document.querySelector(".twitter-btn");
const pinterestBtn = document.querySelector(".pinterest-btn");
const linkedinBtn = document.querySelector(".linkedin-btn");
const whatsappBtn = document.querySelector(".whatsapp-btn");
const mapsBtn = document.querySelector(".maps-btn");


function init() {
  const img = document.querySelector(".img");
  let postUrl = encodeURI(document.location.href);
  let postTitle = encodeURI(document.getElementById("title").textContent);
  let postImg = encodeURI(img.src);
  let location = encodeURI(document.getElementById("location").textContent);
  facebookBtn.setAttribute(
    "href",
    https://www.facebook.com/sharer.php?u=${postUrl}
  );
twitterBtn.setAttribute(
    "href",
    https://twitter.com/share?url=${postUrl}&text=${postTitle}
  );
```

```
pinterestBtn.setAttribute(
    "href",
https://pinterest.com/pin/create/bookmarklet/?media=${postImg}&url=${postUrl}&description=
${postTitle}
  );
  linkedinBtn.setAttribute(
    "href",
    https://www.linkedin.com/shareArticle?url=${postUrl}&title=${postTitle}
  );
  whatsappBtn.setAttribute(
    "href",
    https://wa.me/?text=${postTitle} ${postUrl}
  );
  mapsBtn.setAttribute(
    "href",
    https://www.google.com/maps?q=${location}&ie=UTF8
  );
}
init();
```

The Flask code interacts with an IBM Db2 database to create a web application for displaying and viewing blog posts. This code exemplifies how to establish a connection to the database, retrieve data, and render web pages using the Flask framework. Let's break down the code and its functionality step by step:

1. Imports:

- 'from flask import Flask, render_template` and `import ibm_db` are import statements used to bring in the necessary modules for the application.
- `Flask` is imported from the Flask framework for creating a web application.
- `render_template` is used to render HTML templates in the Flask application.
- ibm_db` is imported to enable the application to interact with the IBM Db2 database.

2. Application Initialization:

```
app = Flask(__name__)
```

- An instance of the Flask application is created. This instance is used to configure and run the web application.

3. Database Connection and Data Retrieval:

```
def get_posts_from_db():

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=31249;UID=qvm12834;
PWD=y6HwghDRDHiUlF1f;", "", "")

stmt = ibm_db.exec_immediate(conn, "SELECT * FROM posts")

posts = []

while ibm_db.fetch_row(stmt):

# Extract post data from the result set and append it to the posts list.

post = {

"heading": ibm_db.result(stmt, "HEADING"),

"location": ibm_db.result(stmt, "LOCATION"),

"image_url": ibm_db.result(stmt, "IMAGE_URL"),

"content": ibm_db.result(stmt, "CONTENT"),

}

posts.append(post)

ibm_db.close(conn)

return posts
```

- The `get_posts_from_db` function is responsible for connecting to the IBM Db2 database and retrieving the list of posts.
- It establishes a database connection using connection details such as the database name, hostname, port, username, and password.
- It executes an SQL query to select all posts from the database table and fetches the results.
- The results are processed row by row, and the data is extracted into a dictionary for each post. These dictionaries are collected into a list called `posts`.
- Finally, the database connection is closed, and the list of posts is returned.

4. Route: Index Page:

```
@app.route('/')

def index():

posts = get_posts_from_db()

return render_template('index.html', posts=posts)
```

- This route is for the index or home page of the application. It is the default landing page when the root URL is accessed.
- It calls the `get_posts_from_db' function to fetch the list of posts from the database.
- The retrieved posts are passed to the `render_template' function, which renders an HTML template named 'index.html' with the posts as data.

5. Route: View Post Page:

```
@app.route('/post/<int:post_id>')

def view_post(post_id):

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=31249;UID=qvm12834;
PWD=y6HwghDRDHiUlF1f;", "", "")

stmt = ibm_db.prepare(conn, "SELECT * FROM posts WHERE ID = ?")

ibm_db.bind_param(stmt, 1, post_id)

if ibm_db.execute(stmt):

row = ibm_db.fetch_assoc(stmt)

ibm_db.close(conn)

if row:

post = {

"heading": row["HEADING"],

"location": row["LOCATION"],
```

```
"image_url": row["IMAGE_URL"],

"content": row["CONTENT"],

}

return render_template('post.html', post=post)

return "Post not found"
```

- This route is used to display an individual post based on its ID, which is passed as a URL parameter.
- It connects to the database and prepares an SQL statement to retrieve the post with the specified ID.
- The `ibm_db.bind_param` method is used to bind the post ID to the SQL statement.
- If the query returns a result, the data is extracted into a dictionary and passed to the 'post.html' template for rendering.
- If no matching post is found, a "Post not found" message is displayed.

6. Application Execution:

```
if __name__ == '__main__':

app.run(debug=True)
```

- The code block inside this condition is executed only if the script is run directly (not imported as a module).
- It starts the Flask application in debug mode, allowing for live code changes and debugging during development.

app.py

```
from flask import Flask, render_template

import ibm_db

app = Flask(__name__)

def get_posts_from_db():

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=31249;UID=qvm12834;
PWD=y6HwghDRDHiUlF1f;", "", "")

stmt = ibm_db.exec_immediate(conn, "SELECT * FROM posts")

posts = []

while ibm_db.fetch_row(stmt):

post = {

"heading": ibm_db.result(stmt, "HEADING"),

"location": ibm_db.result(stmt, "LOCATION"),

"image_url": ibm_db.result(stmt, "IMAGE_URL"),

"content": ibm_db.result(stmt, "CONTENT"),

}

posts.append(post)

ibm_db.close(conn)

return posts

@app.route('/')

def index():

posts = get_posts_from_db()

return render_template('index.html', posts=posts)

@app.route('/post/<int:post_id>')
```

```
def view_post(post_id):

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=31249;UID=qvm12834;
PWD=y6HwghDRDHiUlF1f;", "", "")

stmt = ibm_db.prepare(conn, "SELECT * FROM posts WHERE ID = ?")

ibm_db.bind_param(stmt, 1, post_id)

if ibm_db.execute(stmt):

row = ibm_db.fetch_assoc(stmt)

ibm_db.close(conn)

if row:

post = {

"heading": row["HEADING"],

"location": row["LOCATION"],

"image_url": row["IMAGE_URL"],

"content": row["CONTENT"],

}

return render_template('post.html', post=post)

return "Post not found"

if __name__ == '__main__':

app.run(debug=True)
```

The `requirements.txt` file is a common configuration file used in Python projects, particularly in web applications like Flask, to specify the external packages and their versions that are required for the application to function correctly. Here's an explanation of the contents of this `requirements.txt` file:

1. Flask==2.0.1:

- This line specifies a Python package named "Flask" and a specific version number, "2.0.1," which is required for the application.
- Flask is a lightweight web framework for Python that is commonly used for building web applications. It provides tools and libraries to simplify tasks such as routing, handling HTTP requests and responses, and rendering templates.
- Specifying the version number is important to ensure that the application uses a known and compatible version of Flask. It helps prevent unexpected behaviour or breaking changes that might occur when newer versions of Flask are released.

2. ibm_db==3.0.2:

- This line specifies a Python package named "ibm_db" and a specific version number, "3.0.2," which is required for the application.
- "ibm_db" is a Python driver for IBM Db2 databases. It allows the application to connect to and interact with IBM Db2 databases.
- Just like with Flask, specifying the version of "ibm_db" ensures compatibility and stability. It guarantees that the application uses a version that has been tested and known to work with the specific dependencies and features required for the project.

## requirements.txt

```
Flask==2.0.1

ibm_db==3.0.2
```

1. Base Image Selection:

```
FROM python:3.9
```

This line specifies the base image for the Docker container. In this case, it's using an official Python 3.9 image as the parent image. The official Python images are well-maintained and come with Python pre-installed.

2. Working Directory:

```
WORKDIR /app
```

This line sets the working directory in the container to `/app`. This is where the application code and any subsequent operations will be executed.

3. Copy Application Files:

```
COPY . /app
```

This command copies the contents of the current directory (the directory where the Dockerfile is located) into the `/app` directory in the container. It effectively transfers the application code and files into the container.

4. Install Dependencies:

```
RUN pip install -r requirements.txt
```

This command installs the Python packages listed in the `requirements.txt` file. These are the dependencies required for the application to run. This step is essential to ensure that all necessary packages are available in the container.

5. Expose Port:

```
EXPOSE 5000
```

This line instructs Docker to expose port 5000. It doesn't publish the port to the host; it's more of a declaration that the container will be listening on port 5000. It's up to the user when running the container to map the exposed container port to a host port.

6. Environment Variable:

```
ENV FLASK_APP=app.py
```

This sets an environment variable `FLASK_APP` to `app.py`. This environment variable is used by Flask to determine the main application file. It's a configuration detail for the Flask application.

7. Run Command:

```
CMD ["flask", "run", "--host=0.0.0.0"]
```

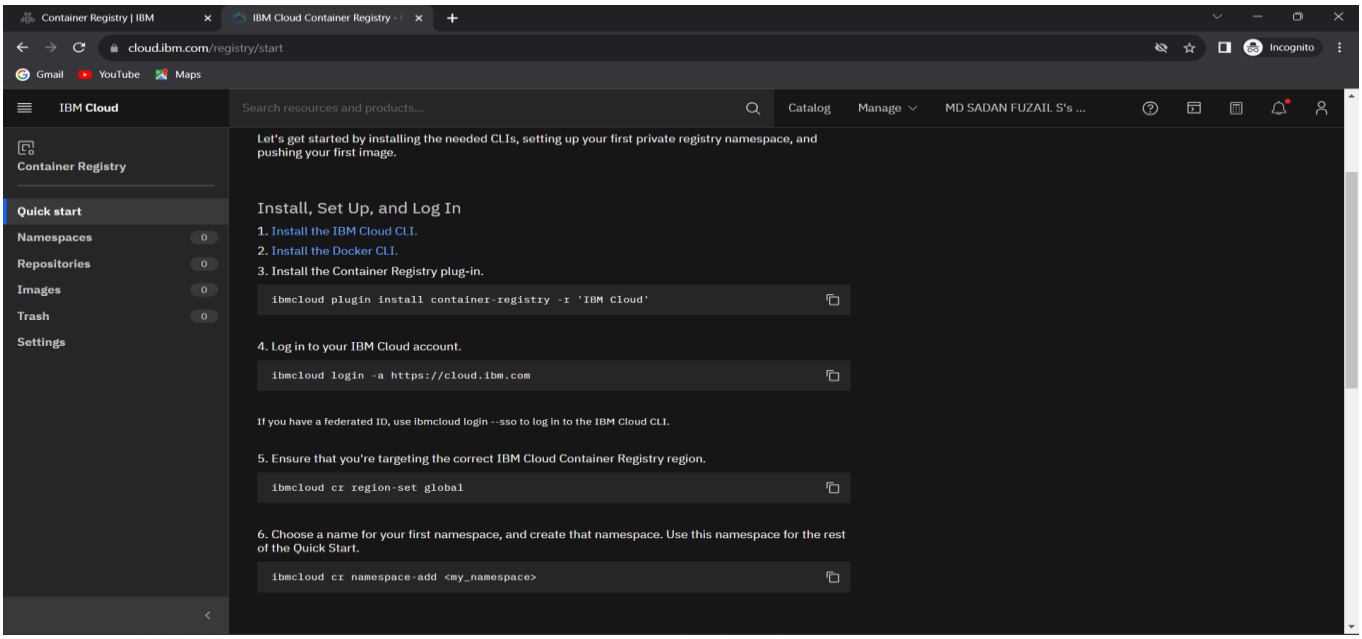
This is the command that will be executed when the container is run. It runs the Flask application using the `flask run` command and specifies `--host=0.0.0.0` to allow external access to the application. The Flask development server will be accessible on port 5000 inside the container.

## Dockerfile

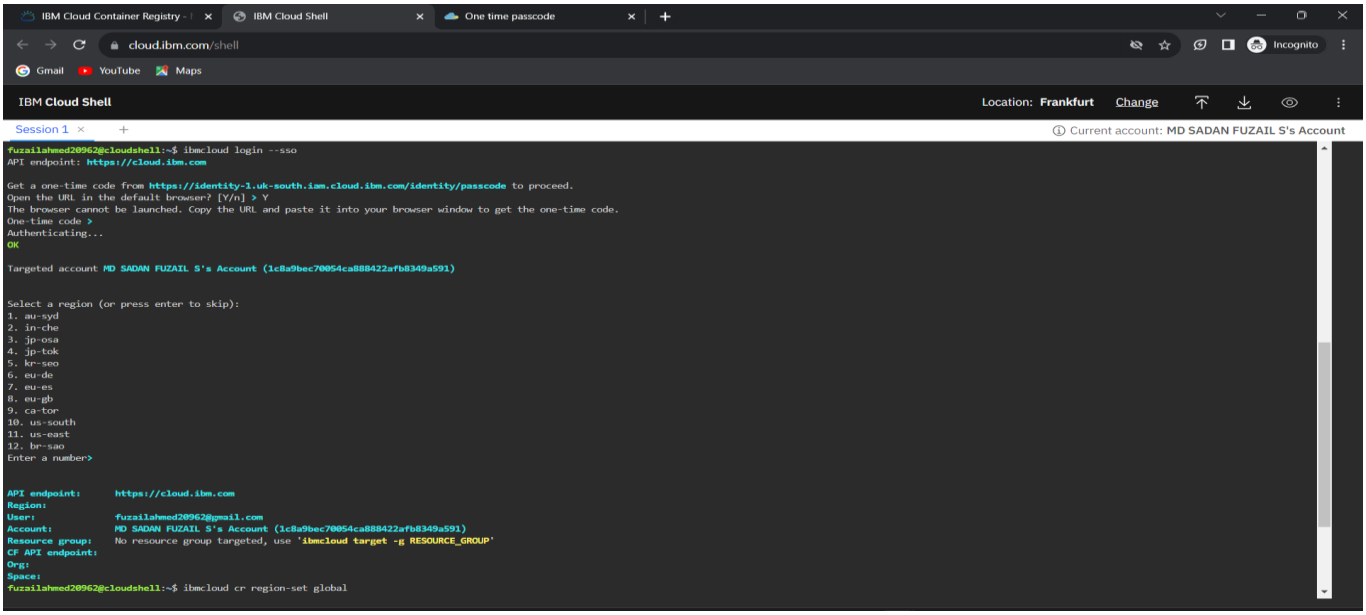
```
# Use an official Python runtime as a parent image

FROM python:3.9

# Set the working directory in the container

WORKDIR /app

# Copy the current directory contents into the container at /app

COPY. /app

# Install any needed packages specified in requirements.txt

RUN pip install -r requirements.txt

# Expose port 5000 for the Flask app

EXPOSE 5000

# Define environment variable for Flask

ENV FLASK_APP=app.py

# Run the Flask app

CMD ["flask", "run", "--host=0.0.0.0"]
```

In the below steps we will be seeing how to deploy our application or code in cloud using Kubernetes. Kubernetes is used to deploy the code from the local machine into the cloud. To deploy the code in Kubernetes, we will use container registry.

Open the container registry and select Quick start, in IBM cloud there will be a step-by-step procedure and commands for execution.




Follow the above steps and enter the commands in IBM cloud shell.

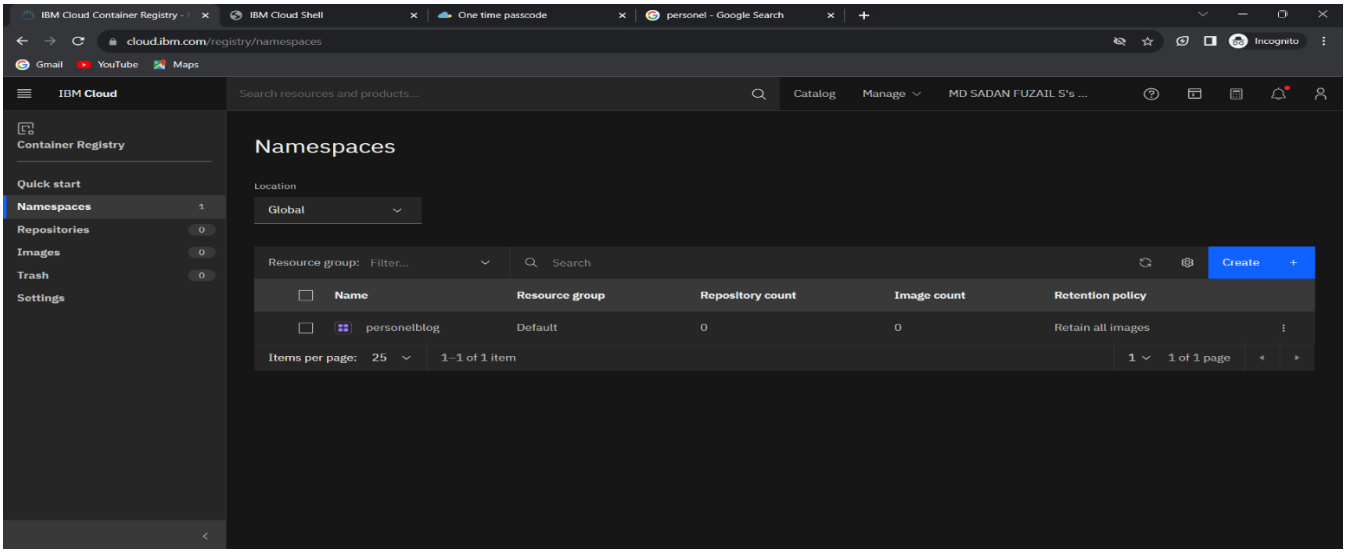



Install the container registry plugin by using the command, **“ibmcloud plugin install container-registry -r 'IBM Cloud'”**.

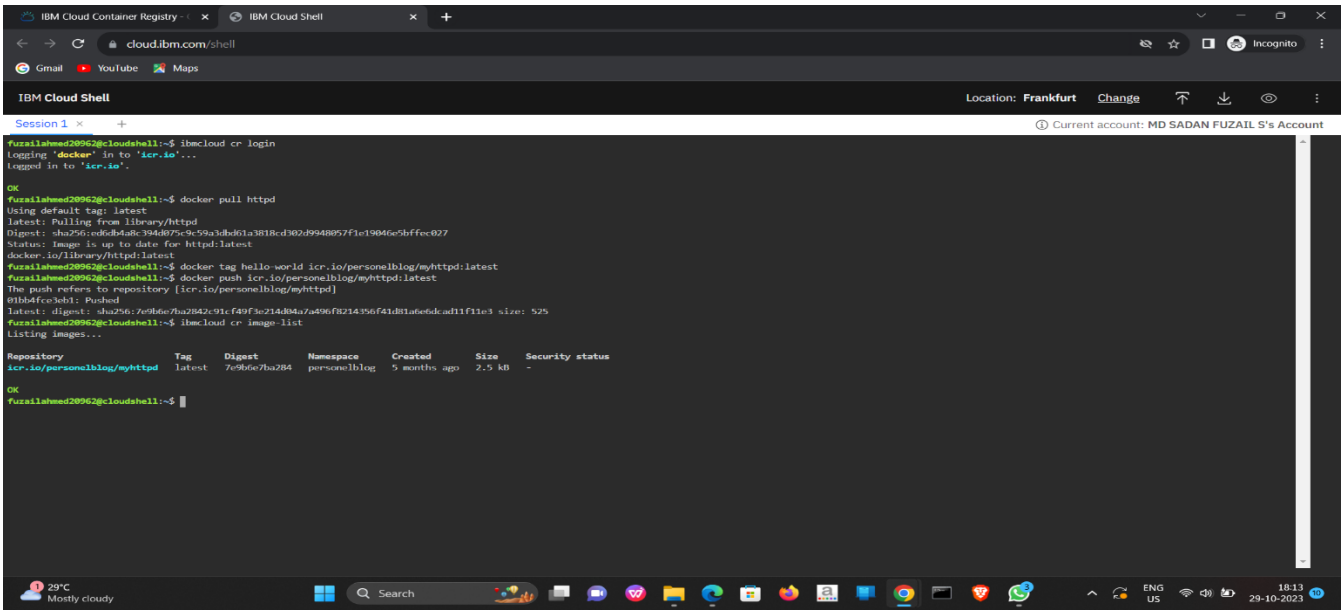
And then Log in to your IBM Cloud account using the command **“ibmcloud login –a https://cloud.ibm.com”** or **“ibmcloud login –sso"** if the first command doesn’t work.

Target the correct IBM Cloud Container Registry region. Use the command “ibmcloud cr region-set global”.

Choose a name and create that namespace by using the command **“ibmcloud cr namespace-add personal blog”**.



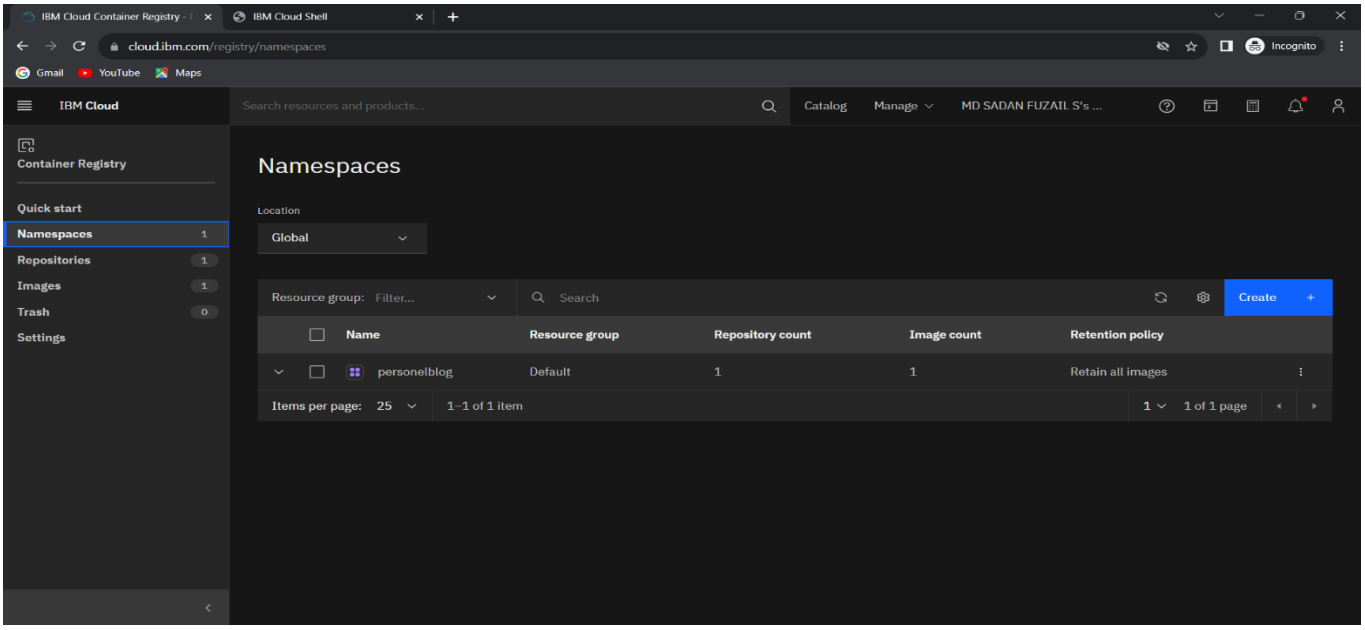

As we can see in the above image the namespace is successfully created.

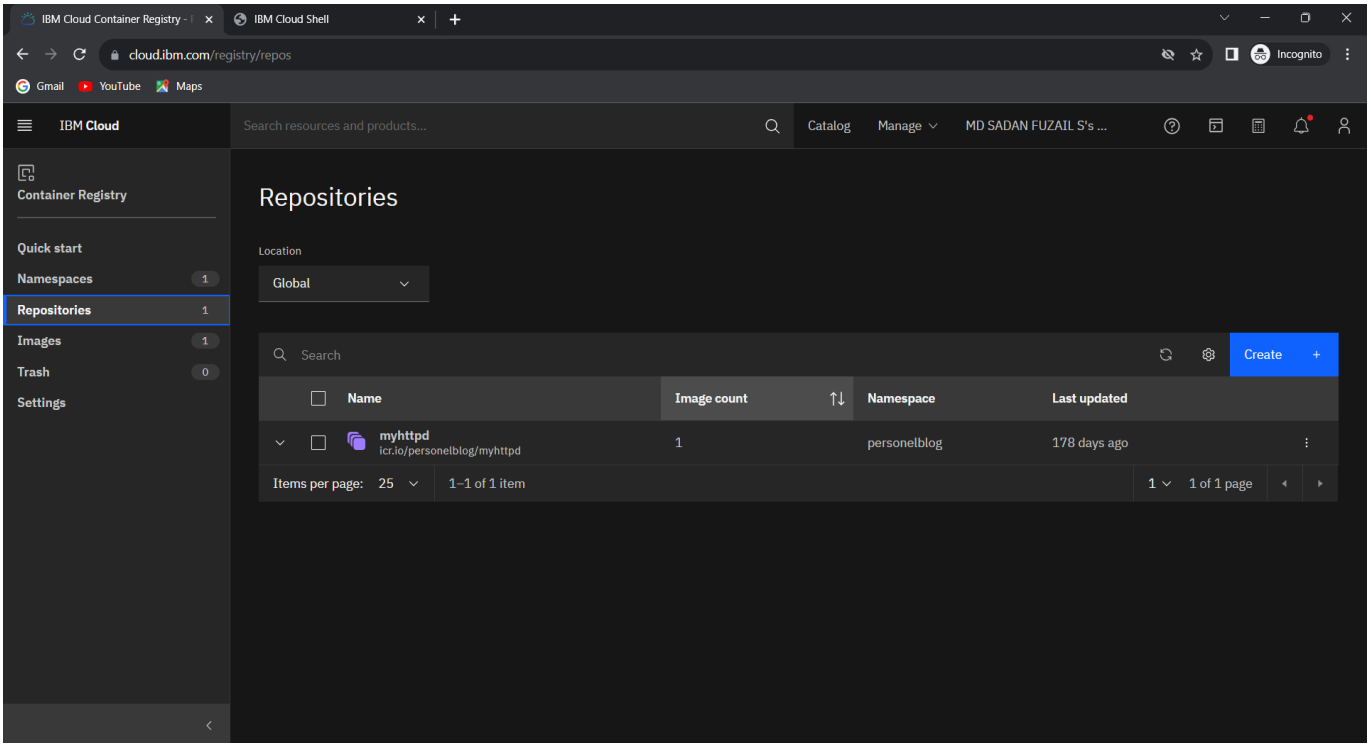


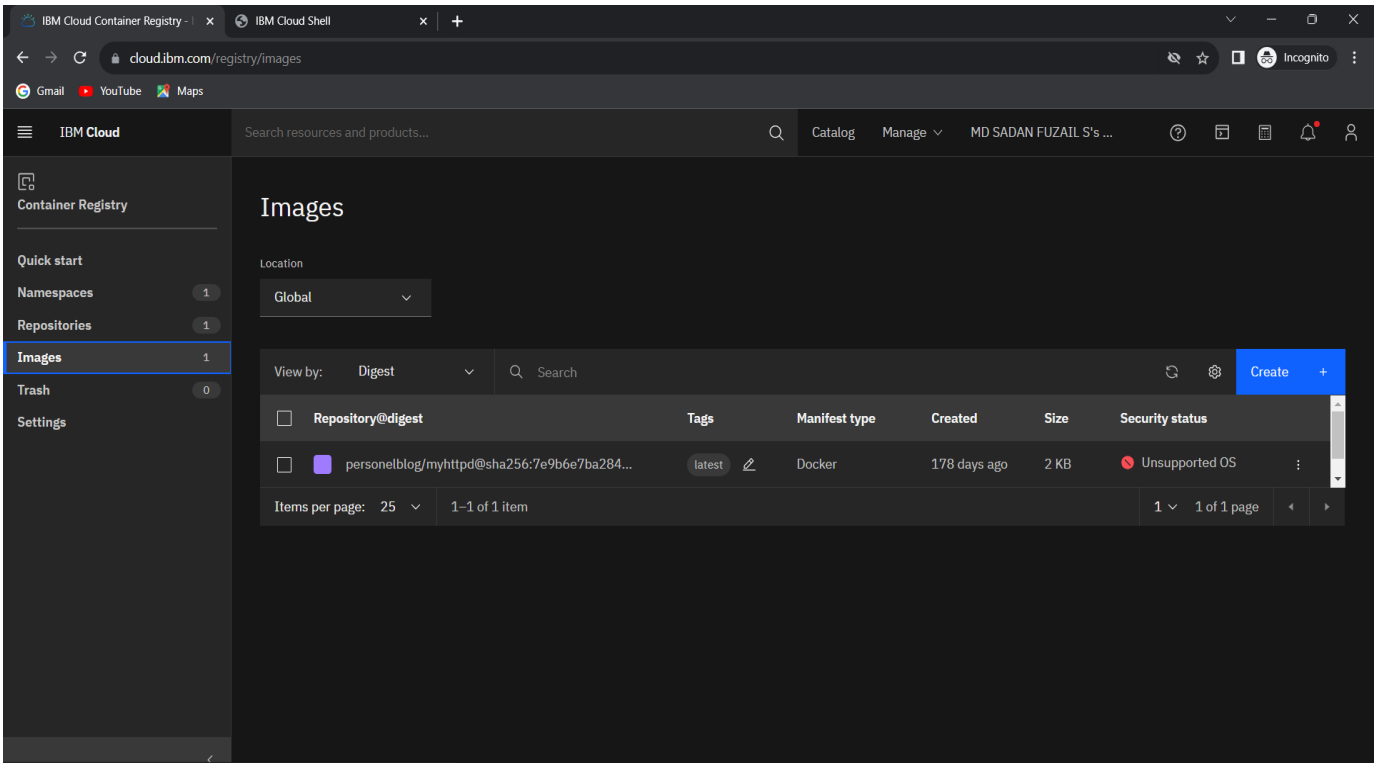

To push the namespace, login in the IBM cloud shell using this command **“ibmcloud cr login”**.
And use the following commands.

1. **docker pull httpd**
2. **docker tag hello-world icr.io/personalblog/myhttpd:latest**
3. **docker push icr.io/personalblog/myhttpd:latest**

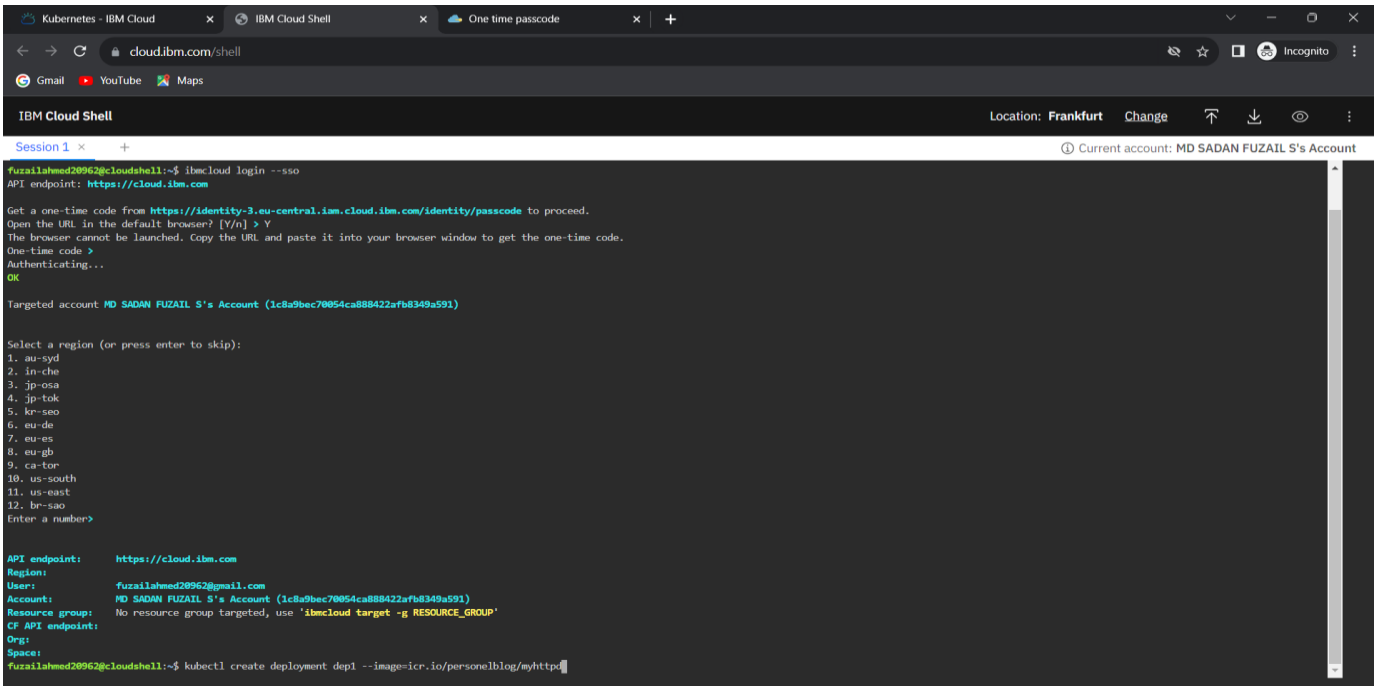
And the namespace will be pushed into the repository. For checking type the command **“ibmcloud cr image-list"** it will show all the pushed namespaces as can be seen in the above image.



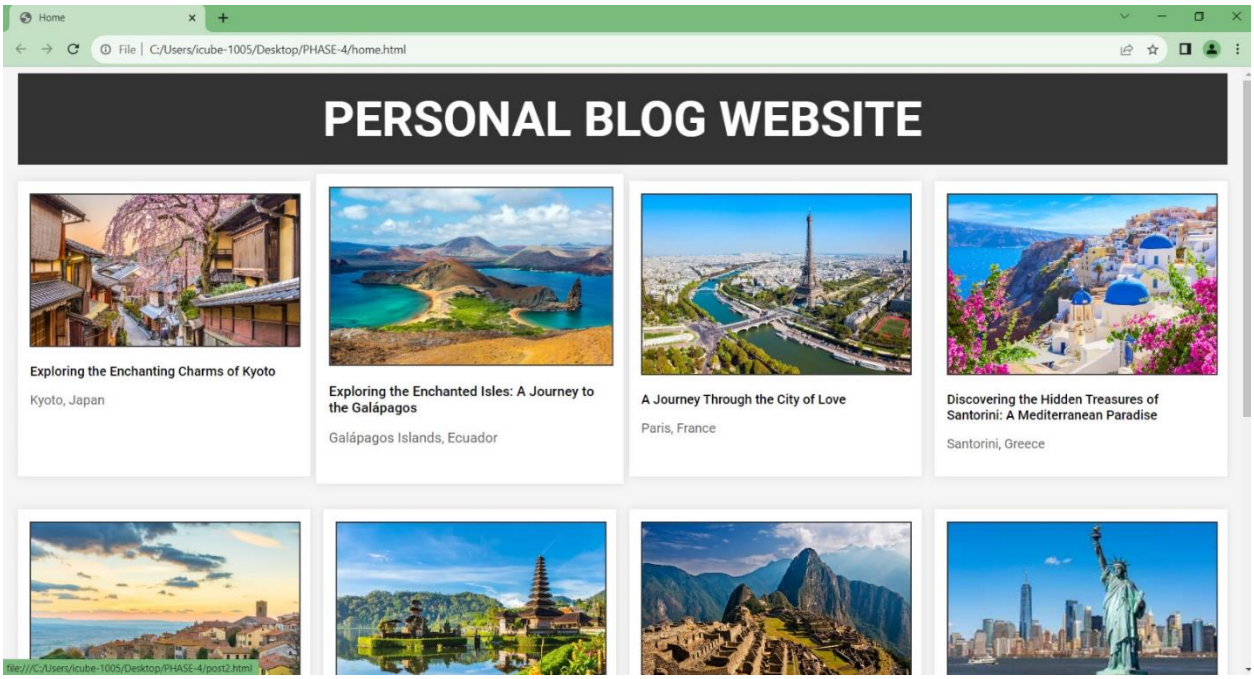


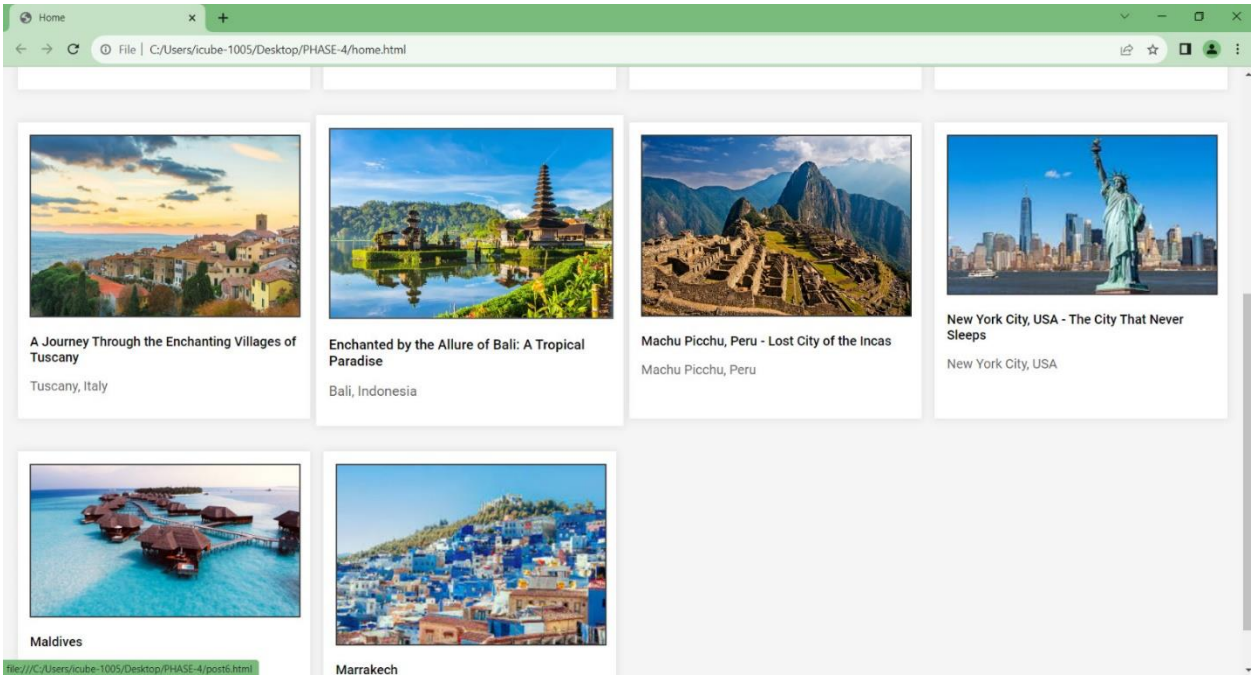

The above images show that the namespace ‘personalblog’ is successfully pushed into the repository.




After pushing the 'personalblog' into the container registry repository, we need to create Kubernetes cluster to deploy the image. Use the command **"kubectl create deployment dep1 --image=icr.io/personalblog/myhttpd"**. This can be done only after upgrading Kubernetes.

By completing the procedure mentioned above, we will obtain the output displayed below.



Charms of Kyoto

File | D:/PHASE-4/post.html

InPrivate

# Exploring the Enchanting Charms of Kyoto

*Kyoto, Japan*

When it comes to captivating travel destinations, Kyoto stands out as a city that effortlessly blends ancient traditions with modern marvels. Nestled in the heart of Japan, Kyoto offers a unique glimpse into the country's rich cultural heritage. From majestic temples like Kinkaku-ji, adorned in shimmering gold leaf, to the serene beauty of Arashiyama Bamboo Grove, this city is a traveler's dream. One can't visit Kyoto without experiencing the exquisite tea ceremonies, where time seems to stand still amidst the tranquil surroundings of traditional tea houses. The Gion district is your gateway to the elusive world of geishas and maikos, while Fushimi Inari Shrine's thousand vermilion torii gates make for a breathtaking hike with panoramic views. Don't forget to savor the city's culinary delights, from delectable kaiseki dinners to street food in Nishiki Market. Kyoto's cherry blossoms in spring and fiery maple leaves in autumn add a touch of natural beauty to an already enchanting city. In Kyoto, every corner unveils a new story, a new tradition, and a new memory waiting to be cherished. So pack your bags and set forth on an unforgettable journey through this cultural gem of Japan.

A Journey to the Galápagos

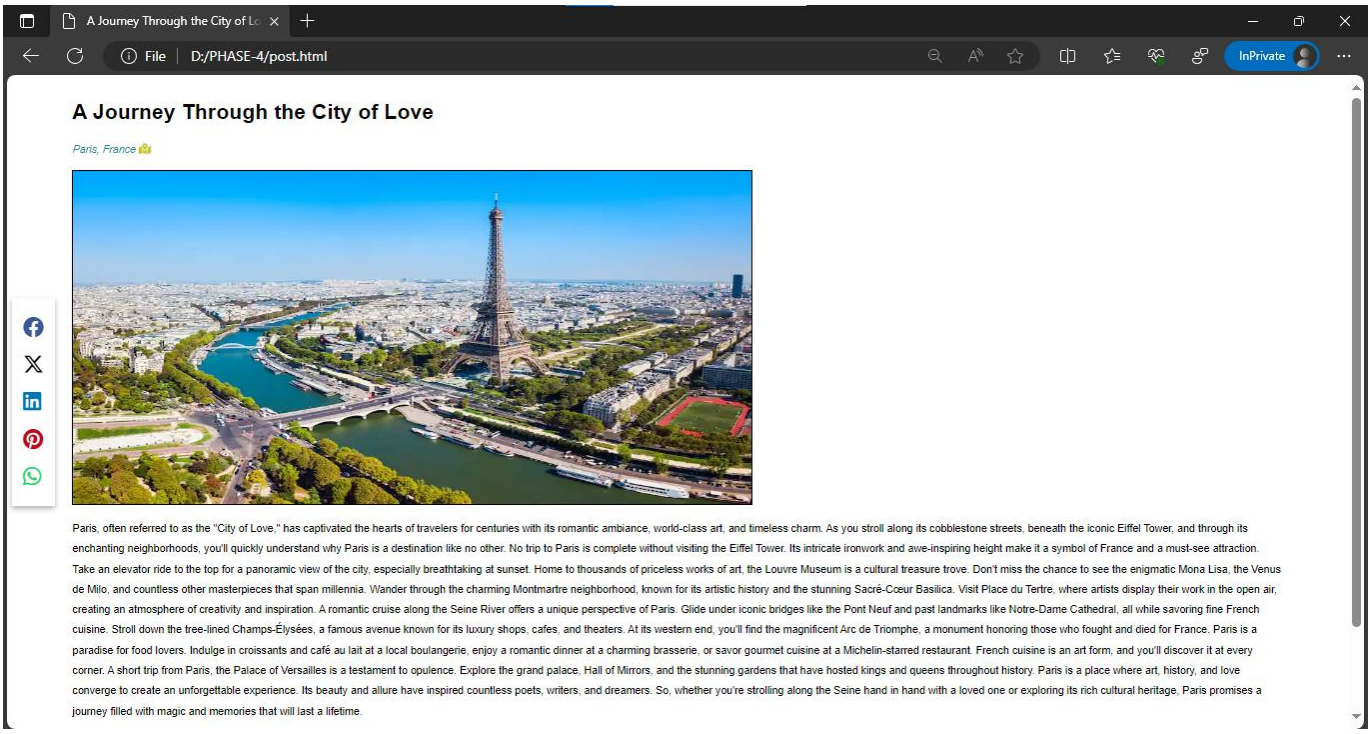
File | D:/PHASE-4/post.html

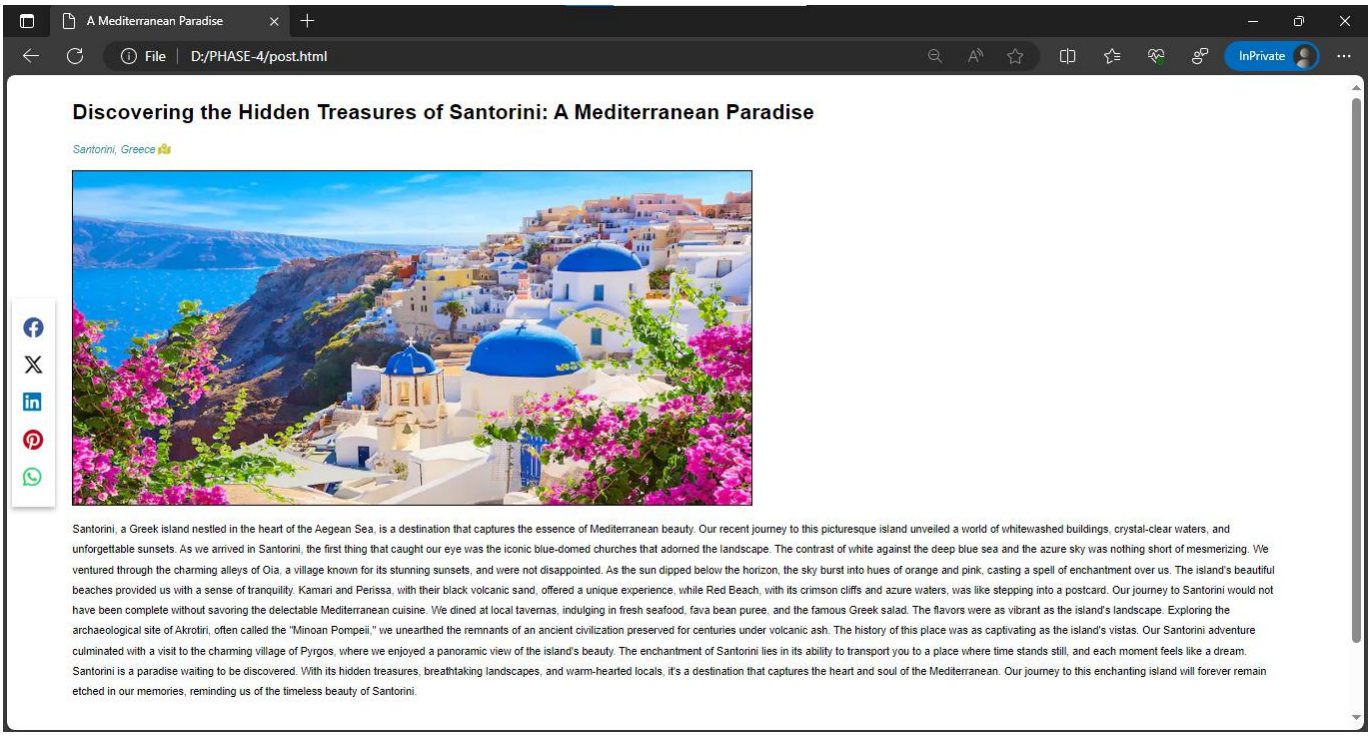
InPrivate

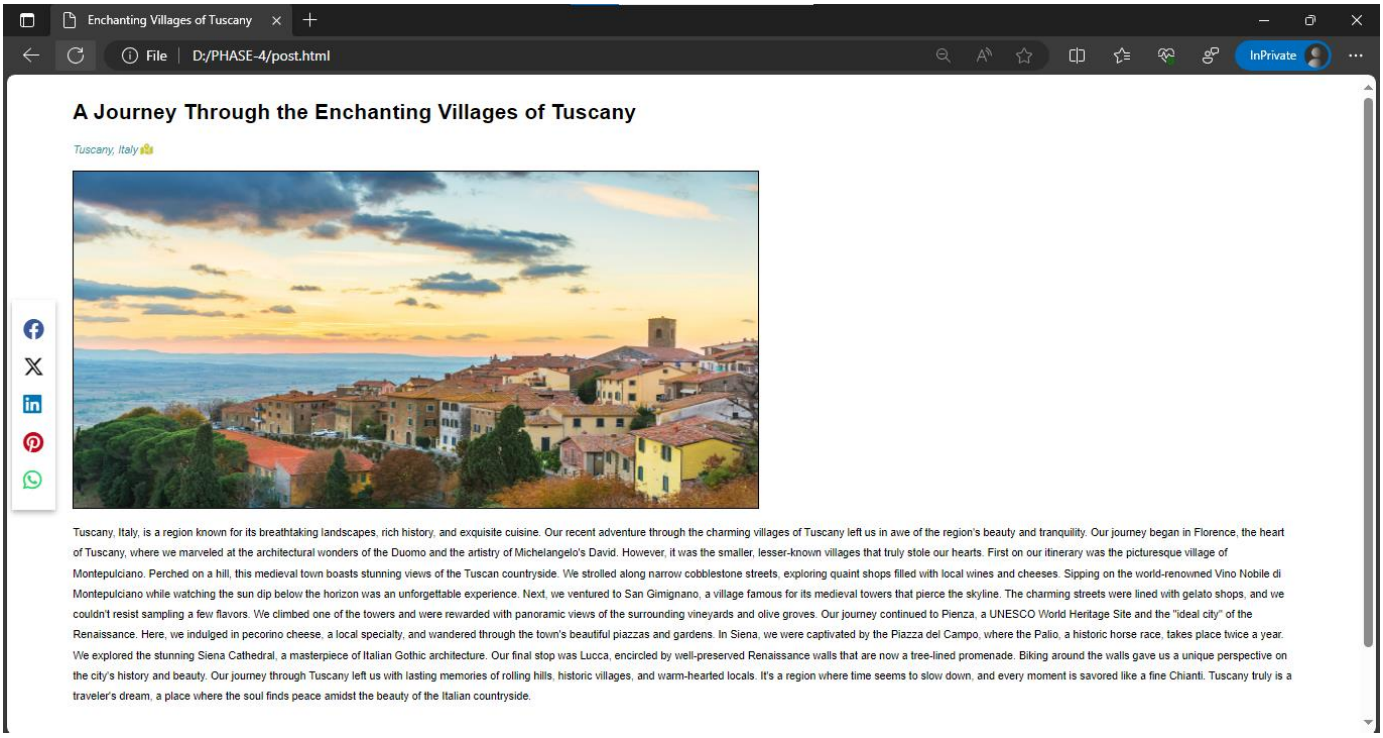
# Exploring the Enchanted Isles: A Journey to the Galápagos

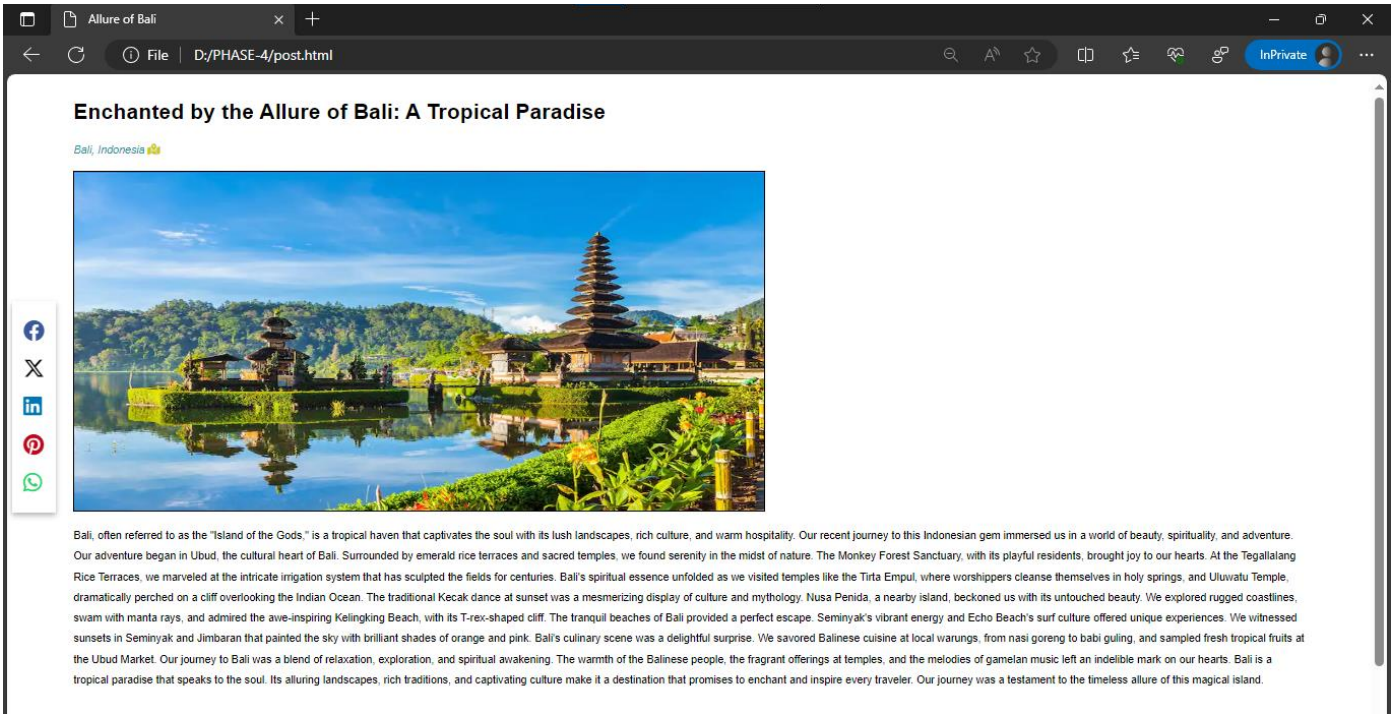
*Galápagos Islands, Ecuador*

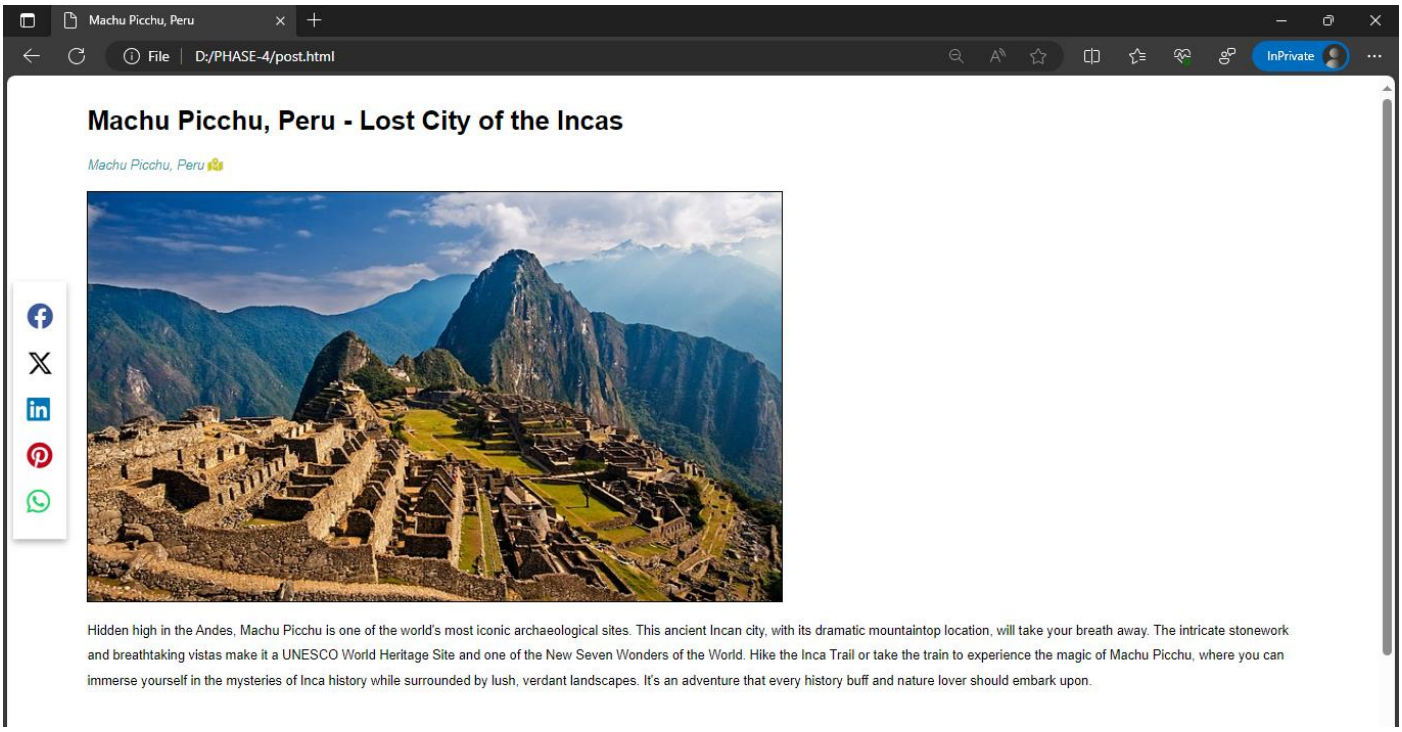
Nestled in the heart of the Pacific Ocean, the Galápagos Islands are a natural wonderland that has captured the imagination of travelers and scientists alike for centuries. This remote archipelago, known for its unique wildlife and pristine landscapes, is a dream destination for any nature enthusiast. Join me on a virtual journey as we explore the hidden gems of the Galápagos in this travel blog post. The Galápagos Islands are home to some of the world's most remarkable and fearless wildlife. From lumbering giant tortoises to playful sea lions, and from curious marine iguanas to colorful bird species like the blue-footed booby, the islands are a living laboratory of evolution. Snorkeling with sea turtles and swimming alongside schools of vibrant fish is an unforgettable experience that immerses you in this unique ecosystem. The Galápagos Islands were formed by volcanic activity millions of years ago, and their rugged, otherworldly landscapes are a testament to this history. Hiking up to the rim of a dormant volcano offers panoramic views of the archipelago, while lava tunnels and rugged terrain provide opportunities for adventurous exploration. The Galápagos Islands are a UNESCO World Heritage Site and a hotspot for conservation efforts. Local authorities and organizations have worked tirelessly to protect and preserve the delicate balance of this ecosystem. Visitors can witness these efforts firsthand by visiting research centers and engaging with passionate conservationists. Traveling responsibly is crucial to preserving the Galápagos' fragile environment. Eco-friendly lodges, tour operators, and strict regulations ensure that your visit has a minimal impact on the delicate ecosystems. Respect for nature and adherence to guidelines are the keys to keeping this paradise pristine for future generations. The Galápagos Islands offer a once-in-a-lifetime opportunity to connect with nature in its purest form. Whether you're a wildlife enthusiast, an adventure seeker, or simply someone who craves the serenity of untouched landscapes, the Galápagos Islands will leave an indelible mark on your soul. Embrace the enchantment of these islands, and you'll find yourself forever captivated by their beauty and biodiversity.


A Journey Through the City of Lo
File | D:/PHASE-4/post.html
InPrivate
A Journey Through the City of Love
Paris, France
Paris, often referred to as the "City of Love," has captivated the hearts of travelers for centuries with its romantic ambiance, world-class art, and timeless charm. As you stroll along its cobblestone streets, beneath the iconic Eiffel Tower, and through its enchanting neighborhoods, you'll quickly understand why Paris is a destination like no other. No trip to Paris is complete without visiting the Eiffel Tower. Its intricate ironwork and awe-inspiring height make it a symbol of France and a must-see attraction. Take an elevator ride to the top for a panoramic view of the city, especially breathtaking at sunset. Home to thousands of priceless works of art, the Louvre Museum is a cultural treasure trove. Don't miss the chance to see the enigmatic Mona Lisa, the Venus de Milo, and countless other masterpieces that span millennia. Wander through the charming Montmartre neighborhood, known for its artistic history and the stunning Sacré-Cœur Basilica. Visit Place du Tertre, where artists display their work in the open air, creating an atmosphere of creativity and inspiration. A romantic cruise along the Seine River offers a unique perspective of Paris. Glide under iconic bridges like the Pont Neuf and past landmarks like Notre-Dame Cathedral, all while savoring fine French cuisine. Stroll down the tree-lined Champs-Élysées, a famous avenue known for its luxury shops, cafes, and theaters. At its western end, you'll find the magnificent Arc de Triomphe, a monument honoring those who fought and died for France. Paris is a paradise for food lovers. Indulge in croissants and café au lait at a local boulangerie, enjoy a romantic dinner at a charming brasserie, or savor gourmet cuisine at a Michelin-starred restaurant. French cuisine is an art form, and you'll discover it at every corner. A short trip from Paris, the Palace of Versailles is a testament to opulence. Explore the grand palace, Hall of Mirrors, and the stunning gardens that have hosted kings and queens throughout history. Paris is a place where art, history, and love converge to create an unforgettable experience. Its beauty and allure have inspired countless poets, writers, and dreamers. So, whether you're strolling along the Seine hand in hand with a loved one or exploring its rich cultural heritage, Paris promises a journey filled with magic and memories that will last a lifetime.


A Mediterranean Paradise
File | D:/PHASE-4/post.html
InPrivate
Discovering the Hidden Treasures of Santorini: A Mediterranean Paradise
Santorini, Greece
Santorini, a Greek island nestled in the heart of the Aegean Sea, is a destination that captures the essence of Mediterranean beauty. Our recent journey to this picturesque island unveiled a world of whitewashed buildings, crystal-clear waters, and unforgettable sunsets. As we arrived in Santorini, the first thing that caught our eye was the iconic blue-domed churches that adorned the landscape. The contrast of white against the deep blue sea and the azure sky was nothing short of mesmerizing. We ventured through the charming alleys of Oia, a village known for its stunning sunsets, and were not disappointed. As the sun dipped below the horizon, the sky burst into hues of orange and pink, casting a spell of enchantment over us. The island's beautiful beaches provided us with a sense of tranquility. Kamari and Perissa, with their black volcanic sand, offered a unique experience, while Red Beach, with its crimson cliffs and azure waters, was like stepping into a postcard. Our journey to Santorini would not have been complete without savoring the delectable Mediterranean cuisine. We dined at local tavernas, indulging in fresh seafood, fava bean puree, and the famous Greek salad. The flavors were as vibrant as the island's landscape. Exploring the archaeological site of Akrotiri, often called the "Minoan Pompeii," we unearthed the remnants of an ancient civilization preserved for centuries under volcanic ash. The history of this place was as captivating as the island's vistas. Our Santorini adventure culminated with a visit to the charming village of Pyrgos, where we enjoyed a panoramic view of the island's beauty. The enchantment of Santorini lies in its ability to transport you to a place where time stands still, and each moment feels like a dream. Santorini is a paradise waiting to be discovered. With its hidden treasures, breathtaking landscapes, and warm-hearted locals, it's a destination that captures the heart and soul of the Mediterranean. Our journey to this enchanting island will forever remain etched in our memories, reminding us of the timeless beauty of Santorini.


Enchanting Villages of Tuscany
File | D:/PHASE-4/post.html
InPrivate
A Journey Through the Enchanting Villages of Tuscany
Tuscany, Italy
Tuscany, Italy, is a region known for its breathtaking landscapes, rich history, and exquisite cuisine. Our recent adventure through the charming villages of Tuscany left us in awe of the region's beauty and tranquility. Our journey began in Florence, the heart of Tuscany, where we marveled at the architectural wonders of the Duomo and the artistry of Michelangelo's David. However, it was the smaller, lesser-known villages that truly stole our hearts. First on our itinerary was the picturesque village of Montepulciano. Perched on a hill, this medieval town boasts stunning views of the Tuscan countryside. We strolled along narrow cobblestone streets, exploring quaint shops filled with local wines and cheeses. Sipping on the world-renowned Vino Nobile di Montepulciano while watching the sun dip below the horizon was an unforgettable experience. Next, we ventured to San Gimignano, a village famous for its medieval towers that pierce the skyline. The charming streets were lined with gelato shops, and we couldn't resist sampling a few flavors. We climbed one of the towers and were rewarded with panoramic views of the surrounding vineyards and olive groves. Our journey continued to Pienza, a UNESCO World Heritage Site and the "ideal city" of the Renaissance. Here, we indulged in pecorino cheese, a local specialty, and wandered through the town's beautiful piazzas and gardens. In Siena, we were captivated by the Piazza del Campo, where the Palio, a historic horse race, takes place twice a year. We explored the stunning Siena Cathedral, a masterpiece of Italian Gothic architecture. Our final stop was Lucca, encircled by well-preserved Renaissance walls that are now a tree-lined promenade. Biking around the walls gave us a unique perspective on the city's history and beauty. Our journey through Tuscany left us with lasting memories of rolling hills, historic villages, and warm-hearted locals. It's a region where time seems to slow down, and every moment is savored like a fine Chianti. Tuscany truly is a traveler's dream, a place where the soul finds peace amidst the beauty of the Italian countryside.


Allure of Bali
File | D:/PHASE-4/post.html
InPrivate
Enchanted by the Allure of Bali: A Tropical Paradise
Bali, Indonesia
Bali, often referred to as the "Island of the Gods," is a tropical haven that captivates the soul with its lush landscapes, rich culture, and warm hospitality. Our recent journey to this Indonesian gem immersed us in a world of beauty, spirituality, and adventure. Our adventure began in Ubud, the cultural heart of Bali. Surrounded by emerald rice terraces and sacred temples, we found serenity in the midst of nature. The Monkey Forest Sanctuary, with its playful residents, brought joy to our hearts. At the Tegallalang Rice Terraces, we marveled at the intricate irrigation system that has sculpted the fields for centuries. Bali's spiritual essence unfolded as we visited temples like the Tirta Empul, where worshippers cleanse themselves in holy springs, and Uluwatu Temple, dramatically perched on a cliff overlooking the Indian Ocean. The traditional Kecak dance at sunset was a mesmerizing display of culture and mythology. Nusa Penida, a nearby island, beckoned us with its untouched beauty. We explored rugged coastlines, swam with manta rays, and admired the awe-inspiring Kelingking Beach, with its T-rex-shaped cliff. The tranquil beaches of Bali provided a perfect escape. Seminyak's vibrant energy and Echo Beach's surf culture offered unique experiences. We witnessed sunsets in Seminyak and Jimbaran that painted the sky with brilliant shades of orange and pink. Bali's culinary scene was a delightful surprise. We savored Balinese cuisine at local warungs, from nasi goreng to babi guling, and sampled fresh tropical fruits at the Ubud Market. Our journey to Bali was a blend of relaxation, exploration, and spiritual awakening. The warmth of the Balinese people, the fragrant offerings at temples, and the melodies of gamelan music left an indelible mark on our hearts. Bali is a tropical paradise that speaks to the soul. Its alluring landscapes, rich traditions, and captivating culture make it a destination that promises to enchant and inspire every traveler. Our journey was a testament to the timeless allure of this magical island.


Machu Picchu, Peru
File | D:/PHASE-4/post.html
InPrivate
Machu Picchu, Peru - Lost City of the Incas
Machu Picchu, Peru
Hidden high in the Andes, Machu Picchu is one of the world's most iconic archaeological sites. This ancient Incan city, with its dramatic mountaintop location, will take your breath away. The intricate stonework and breathtaking vistas make it a UNESCO World Heritage Site and one of the New Seven Wonders of the World. Hike the Inca Trail or take the train to experience the magic of Machu Picchu, where you can immerse yourself in the mysteries of Inca history while surrounded by lush, verdant landscapes. It's an adventure that every history buff and nature lover should embark upon.


New York City, USA
File | D:/PHASE-4/post.html
InPrivate
New York City, USA - The City That Never Sleeps
New York City, USA
New York City, often referred to as the "Big Apple," is a metropolis that needs no introduction. From the dazzling lights of Times Square to the tranquility of Central Park, New York offers something for everyone. Explore world-class museums like the Met and MoMA, savor diverse culinary delights in neighborhoods like Chinatown and Little Italy, and take in the skyline from the Empire State Building. Broadway shows, iconic landmarks like the Statue of Liberty, and the energetic atmosphere of this urban jungle make New York a destination that lives up to its reputation.

Maldives

File | D:/PHASE-4/post.html

InPrivate

# Maldives

*Maldives*

The Maldives is a paradise for beach lovers and water enthusiasts. This tropical nation, comprising over 1,000 coral islands, offers luxury overwater bungalows, clear blue waters, and vibrant marine life. Snorkeling and diving are must-do activities to explore the colorful coral reefs. Enjoy romantic sunsets, rejuvenating spa treatments, and sumptuous seafood on the sand. Whether you're a honeymooner or just seeking a tranquil escape, the Maldives is the ultimate destination.

Marrakech, Morocco

File | D:/PHASE-4/post.html

InPrivate

# Marrakech

*Marrakech, Morocco*

Marrakech, the Red City, is a vibrant and exotic destination that promises an unforgettable experience. Lose yourself in the bustling souks, where you can haggle for spices, textiles, and handcrafted treasures. Explore the historic medina with its stunning palaces and gardens, such as the Bahia Palace and Jardin Majorelle. Indulge in Moroccan cuisine and enjoy the enchanting atmosphere of the city's lively square, Jemaa el-Fna.

## Conclusion

We built a website to host a personal blog on the IBM Cloud using HTML, CSS, Flask, IBM db2/SQLlite. The document containing all the details regarding the deployment of the website was uploaded. We have used Docker to create an image of our application and push it to the IBM Container Registry. We then deployed the code using Kubernetes, which is a platform for managing containerized workloads and services. We have also tested our website and verified its functionality.

By following the steps in this document, we have successfully created a personal blog website on the IBM Cloud that can be accessed by anyone on the internet. We have also demonstrated our skills in web development, cloud computing, and containerization. In this document we have highlighted how to build and deploy a website on the IBM Cloud using HTML, CSS, Flask, IBM db2/SQLlite, Docker, and Kubernetes.