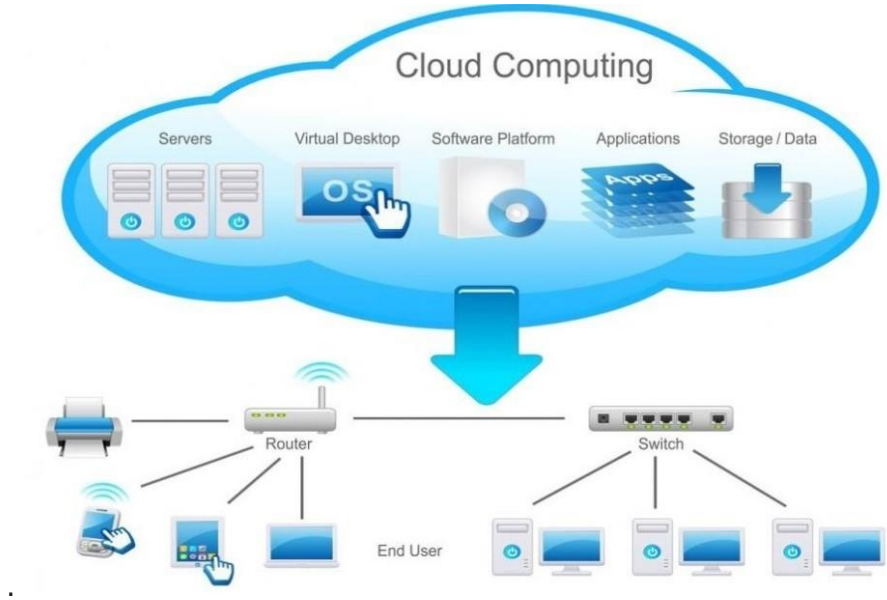
PHASE 2

Project Title: personal Travel blog on IBM Cloud Static Web Apps



Introduction:

The travel blog application can also communicate with the database and storage to store and retrieve data. For example, when a user writes a new blog post, the travel blog application will store the blog post in the database. When the user wants to view a blog post, the travel blog application will retrieve the blog post from the database and display it to the user.

Personalization: The blog can be highly personalized to reflect the unique travel experiences and interests of the blogger. For example, the blogger can use IBM Watson AI services to generate personalized recommendations for destinations, activities, and restaurants.

Engagement: The blog can be used to engage with readers in a variety of ways. For example, the blogger can use IBM Cloud Social Media services to create interactive polls, quizzes, and contests.

Immersion: The blog can be used to create an immersive experience for readers. For example, the blogger can use IBM Cloud Video and Image services to embed videos and photos of their travels in their blog posts.

Here are some specific examples of innovative features that we are going to implemented in a personal travel blog on an IBM Cloud Static Web App:

Interactive travel map: The blog could include an interactive travel map that shows the blogger's route, destinations, and activities. The map could be powered by IBM Cloud Geospatial services.

Personalized travel itineraries: The blog could generate personalized travel itineraries for readers based on their interests and budget. The itineraries could be generated using IBM Watson AI services.

Travel photo gallery: The blog could include a travel photo gallery that showcases the blogger's best travel photos. The photo gallery could be powered by IBM Cloud Image services.

Travel video blog: The blog could include a travel video blog where the blogger shares videos of their travels. The video blog could be powered by IBM Cloud Video services.

Travel social media wall: The blog could include a travel social media wall that displays the blogger's social media posts about their travels. The social media wall could be powered by IBM Cloud Social Media services.

Sample coding:

Home.html:

```
<!DOCTYPE html>
<html>
<head>
   <title>My Travel Blog</title>
```

```
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <h1>Welcome to My Travel Blog</h1>
    <p>This is your homepage. You can navigate to other pages from here.</p>
    <a href="{{ url_for('login') }}">Login</a> | <a href="{{ url_for('dashboard') }}">Dashboard</a>
</body>
</html>
```

dashboard.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Dashboard</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <h1>Dashboard</h1>
    <p>Welcome to your dashboard. You can create and manage your travel blog posts here.</p>
    <a href="{{ url_for('logout') }}">Logout</a>
</body>
</html>
```

style.css

```
body {
    font-family: Arial, sans-serif;
}
h1 {
    color: #333;
}
form {
    width: 300px;
    margin: 0 auto;
}
input {
    display: block;
    margin: 10px 0;
    padding: 5px;
}
```

App.py

```
from flask import Flask, render_template, request, redirect, url_for, session

app = Flask(__name)
app.secret_key = 'your_secret_key'  # Change this to a secret key for your application
```

```
# Sample users (replace with a database in a real application)
users = {
    'user1': 'password1',
    'user2': 'password2',
}

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if users.get(username) == password:
            session['username'] = username
            return redirect(url_for('dashboard'))
        else:
            return 'Invalid username or password'
    return render_template('login.html')

@app.route('/dashboard')
def dashboard():
    if 'username' in session:
        return render_template('dashboard.html')
```

```
    return redirect(url_for('login'))


@app.route('/logout')
def logout():
    session.pop('username', None)
    return redirect(url_for('home'))

if __name__ == '__main__':
    app.run(debug=True)
```