E-COMMERCE APPLICATION ON IBM CLOUD FOUNDRY

PHASE 3: Development Part 1

In this Ecommerce application project we will begin our project using IBM Cloud Foundry. We will perform different functions as per project requirement.

IBM CLOUD FOUNDRY

IBM Cloud Foundry is a platform-as-a-service (PaaS) offering provided by IBM that allows developers to deploy and manage applications in a cloud environment. Here are some key points about IBM Cloud Foundry:

- Open Source Foundation: Cloud Foundry is an open-source PaaS platform, and IBM's offering is based on this open standard, ensuring flexibility and compatibility with a wide range of development tools and languages.

- Application Deployment: It enables developers to easily deploy and manage applications, including web applications and microservices, without having to worry about the underlying infrastructure.

- Multi-Language Support: IBM Cloud Foundry supports multiple programming languages, making it suitable for a variety of development needs. This includes Java, Node.js, Python, Ruby, Go, and more.

- Scaling and Load Balancing: It provides automatic load balancing and horizontal application scaling, allowing your applications to handle increased traffic efficiently.

- Services and Integration: IBM Cloud Foundry integrates with a wide range of cloud services, databases, and tools, making it easier to build, test, and scale applications with the services they need.

- Developer Productivity: Developers can focus on coding and application logic while the platform takes care of infrastructure and operational aspects like patching and scaling.

To create an e-commerce website using HTML, CSS, and JavaScript on IBM Cloud:

1. Create an IBM Cloud Account:
   - Visit [IBM Cloud](https://cloud.ibm.com/registration) and sign up for an IBM Cloud account if you don't have one. You may need to provide payment information, but IBM Cloud offers a free tier with certain limitations.

2. Choose a Cloud Service:
   - IBM Cloud provides various cloud services, including virtual servers and serverless computing options like IBM Cloud Functions. The choice depends on your specific needs and technical expertise.

3. Set Up a Web Server:
   - If you opt for a virtual server, you can create a Virtual Private Cloud (VPC) to isolate your resources. Then, provision a virtual server instance.
   - Configure your server by installing the necessary software components (e.g., web server software like Apache or Nginx), and ensure it's accessible over the internet.

4. Domain Name:
   - You can either purchase a domain name through IBM Cloud or configure an existing domain to point to your IBM Cloud server. This may involve setting up DNS records.

5. Code Your Website:
   - Develop your e-commerce website using HTML, CSS, and JavaScript. You'll create web pages for product listings, shopping cart functionality, and the checkout process. This is where you define your website's layout, appearance, and functionality.

6. Database:
   - Choose a database service on IBM Cloud to store product information, user data, and order history. You can use IBM Db2 for structured data, IBM Cloudant for NoSQL data, or other databases available on the platform.

7. Security:
   - Implement security measures to protect your e-commerce website:
     - Configure SSL certificates to enable HTTPS for secure data transmission.
     - Use IBM Cloud Identity and Access Management (IAM) for access control and user authentication.
     - Ensure your database is secured with proper access controls.

8. Optimization:
   - Optimize your website's performance:
     - Utilize IBM Cloud's Content Delivery Network (CDN) to distribute content globally, reducing latency.
     - Implement caching mechanisms to improve load times.

9. Authentication:
   - Implement user authentication and authorization systems to secure user accounts and control access to specific parts of your website. IBM Cloud IAM can assist with this.

10. Payment Processing:
   - Integrate with a payment gateway service to handle payment processing. You can use third-party services like Stripe or PayPal or explore IBM Cloud's payment processing options.

11. Scaling:
   - Design your architecture with scalability in mind. IBM Cloud offers auto-scaling options, enabling your website to handle increased traffic automatically.

12. Testing and Debugging:
   - Thoroughly test your e-commerce website on IBM Cloud to ensure it works as expected. Debug any issues that may arise, paying attention to any cloud-specific challenges.

13. Deployment:
   - Deploy your website to the IBM Cloud server. You can typically upload your HTML, CSS, and JavaScript files via SCP (Secure Copy Protocol) or use Git for version control.

14. Monitoring:
   - Set up monitoring and analytics using IBM Cloud's monitoring and logging tools. Monitor the performance of your website and gain insights into user behavior.

15. Maintenance:
   - Continuously monitor and update your website. IBM Cloud makes it easy to deploy updates and patches as needed.

16. Backup and Disaster Recovery:
   - Implement regular backups of your data to ensure data safety and develop a disaster recovery plan in case of unexpected issues.

17. Compliance:
   - Ensure that your e-commerce website complies with legal and privacy regulations, such as GDPR or CCPA.

HTML (index.html):

```
<!DOCTYPE html>
<html>
<head>
        <title>My E-Commerce Website</title>
        <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
        <header>
                <h1>Welcome to My E-Commerce Store</h1>
        </header>
        <main>
                <section class="product">
                <img src="product1.jpg" alt="Product 1">
                <h2>Product 1</h2>
                <p>Price: $20</p>
                <button onclick="addToCart('Product 1', 20)">Add to Cart</button>
                </section>
        </main>
        <aside>
                <h2>Shopping Cart</h2>
                <ul id="cart">
                </ul>
        </aside>
        <footer>
                <p>&copy; 2023 My E-Commerce Store</p>
        </footer>
        <script src="script.js"></script>
</body>
</html>
```

CSS (styles.css):

```
.product {
        border: 1px solid #ccc;
        padding: 10px;
        margin: 10px;
        text-align: center;
}
```

JavaScript (script.js):

```
let cart = [];
function addToCart(productName, price) {
        cart.push({ product: productName, price: price });
        updateCart();
}
function updateCart() {
        let cartList = document.getElementById('cart');
        cartList.innerHTML = '';
        let total = 0;
        cart.forEach(item => {
                total += item.price;
                let listItem = document.createElement('li');
                listItem.textContent = `${item.product} - $${item.price}`;
                cartList.appendChild(listItem);
        });
        let totalItem = document.createElement('li');
        totalItem.textContent = `Total: $${total}`;
        cartList.appendChild(totalItem);
}
```

In this simplified example, we have an HTML structure for our e-commerce website, including product listings, a shopping cart, and a basic layout. The CSS file provides some styling, and the JavaScript file handles adding products to the cart and updating the cart's content in real-time.

.