IBM-CAD

PROJECT DESIGN PHASE 1

Dhana Lakshmi

TEAM LEAD : DHANA LAKSHMI S

TEAM M1 : KALPANA SRI R

TEAM M2 : JEEVITHA V

DEPARTMENT OF CSE

KARPAGAM INSTITUTE OF TECHNOLOGY,COIMBATORE

**PROPOSED SYSTEM**

Here's our proposed system for an image resize website using IBM Cloud:

**1. Design the User Interface:**

- Create a user-friendly web interface where users can upload images and specify resizing options.

- Use HTML, CSS, and JavaScript to design the interface.

- Include form fields for selecting dimensions, aspect ratio, file format, and compression level.

**2. Set up IBM Cloud Object Storage (COS):**

- Create an IBM Cloud account

- Create an instance of IBM Cloud Object Storage service.

- Create two buckets: one for storing original images and another for resized images.

**3. Develop Backend Logic:**

- Choose a server-side programming language like Node.js, Python, or Java.

- Use a framework like Express.js or Flask to handle HTTP requests.

- Implement the image resize functionality using image processing libraries (e.g., Sharp for Node.js, Pillow for Python).

- Use the IBM Cloud Object Storage SDK or library to interact with the COS buckets.

**4. Deploy Backend to IBM Cloud:**

- Containerize the backend application using Docker.

- Push the Docker image to IBM Cloud Container Registry or another container registry of your choice.

- Create a Kubernetes cluster on IBM Cloud Kubernetes Service (IKS).

- Deploy the backend application as a Kubernetes Deployment or StatefulSet in the cluster.

**5. Implement File Upload and Processing:**

- Configure the backend application to handle file uploads from the user interface.

- Store the uploaded images in the original images COS bucket.

- Retrieve the uploaded image from COS, resize it based on user-defined options, and save the resized image in the resized images COS bucket.

**6. Configure IBM Cloud CDN:**

- Enable the IBM Cloud Content Delivery Network (CDN) for the resized images COS bucket.

- Set up the necessary configurations to ensure efficient content delivery and caching.

**7. Integrate CDN with Website:**

- Update the user interface to display the resized images using the CDN URL.

- Use HTML `<img>` tags with the appropriate CDN URLs to render the resized images.

**8. Enable Auto Scaling and Monitoring:**

- Configure auto-scaling rules for our Kubernetes cluster based on resource utilization.

- Set up monitoring and logging using IBM Cloud Monitoring and Logging services to track the performance and health of our system.

**9. Implement Security Measures:**

- Implement authentication and authorization mechanisms to protect user data and prevent unauthorized access.

- Apply security best practices such as input validation, sanitization, and encryption to ensure data integrity.

**10. Test and Deploy:**

- Perform thorough testing of the website to ensure proper functionality and responsiveness.

- Deploy the updated website to the IBM Cloud platform using services like IBM Cloud Foundry or IBM Cloud Kubernetes Service.

**11. Continuous Integration and Deployment (CI/CD):**

- Set up a CI/CD pipeline using tools like Jenkins, Travis CI, or IBM Cloud Continuous Delivery services to automate the deployment process.

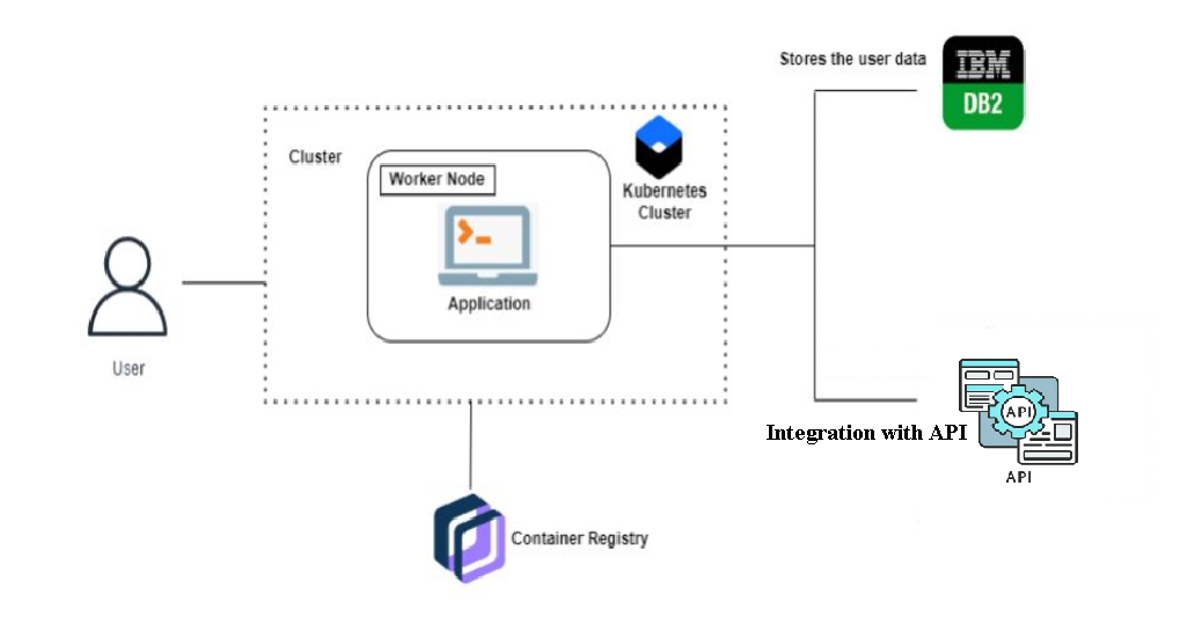
**12. Monitor and Maintain:**

- Continuously monitor the website's performance, user feedback, and system logs.

- Regularly maintain and update the system, including security patches, feature enhancements, and bug fixes.

Remember, this is a high-level overview, and each step may require further configuration and customization based on your specific requirements and preferences.

**SOLUTION ARCHITECTURE**



IBM Cloud – kubernetes - Docker

