CLOUD NATIVE EXPENSE TRACKER USING CI/CD

What is the Cloud Native Expense Tracker?

Definition

A cloud-native website designed to help individuals or organizations monitor and manage their expenses with scalability, reliability, and real-time access across devices.

Architecture

The architecture uses a React.js as frontend, Node.js and Express.js as backend, MongoDB database, containerized with Docker, deployed via Kubernetes, and automated using Jenkins and GitHub

Deployment

Deployed using Docker containers managed by Kubernetes, with automated CI/CD through Jenkins and GitHub.

Why is it Important?

Financial visibility

Provides users with real-time insights into spending patterns, helping to identify areas of overspending and enabling better budgeting.

Scalability

Designed to handle fluctuating user demands smoothly, making it suitable for both individual users and businesses as they grow.

Efficiency

Automates manual data tracking and reporting tasks, saving significant time and reducing human error.

Who Can Benefit From It?

01

Individual users

People looking for a simple and effective way to track expenses, categorize spending, and manage personal budgets.

02

Small businesses

Entrepreneurs who need a cost-friendly solution to manage operations expenses, generate reports, and plan for financial sustainability.

03

Development teams

Those aiming to learn cloud-native technologies and CI/CD implementation can use the platform as a hands-on project.

Technology Stack and Rationale

Frontend

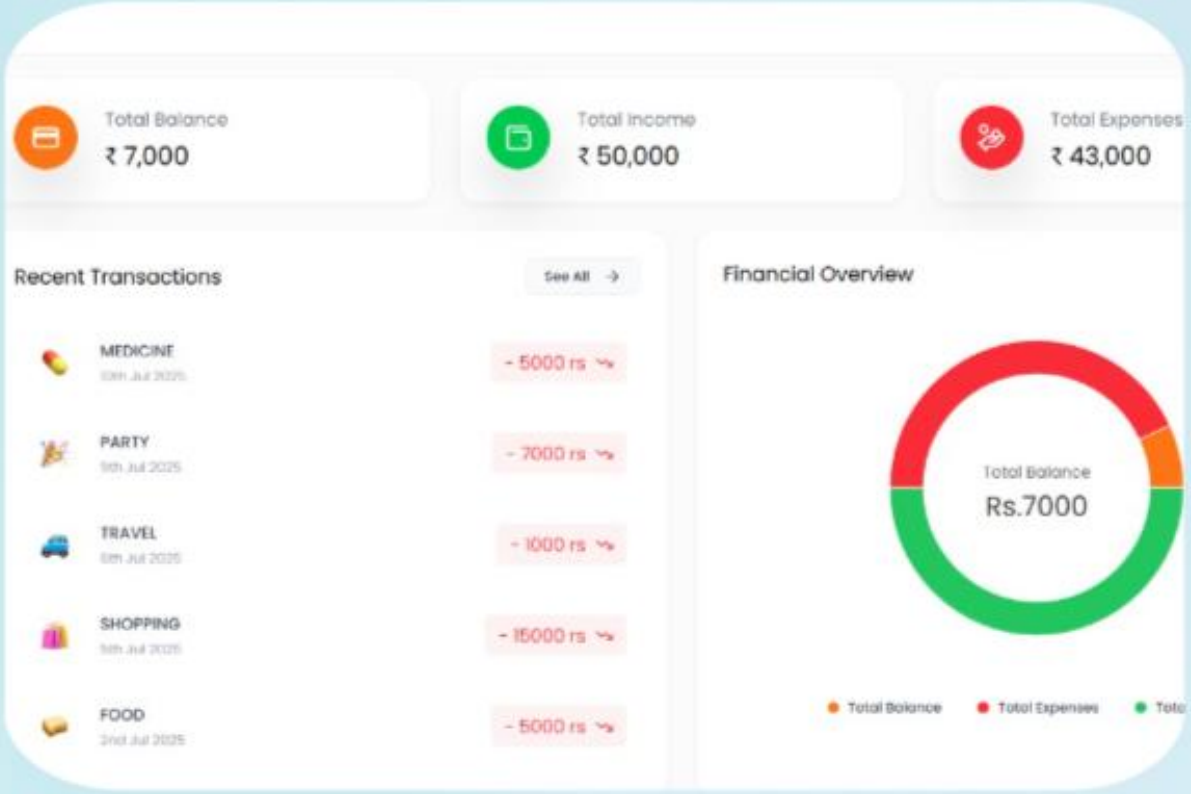
Leverage React for building dynamic and responsive user interfaces with extensive component reusability.

Backend

Use Node.js for a lightweight and scalable server-side logic development.

Database

Incorporate MongoDB for its flexibility in handling unstructured or semi-structured expense data.



Technology Stack and Rationale

01

Continuous Integration

Use GitHub Actions to automate testing and integration of code from multiple contributors.

02

Continuous Deployment

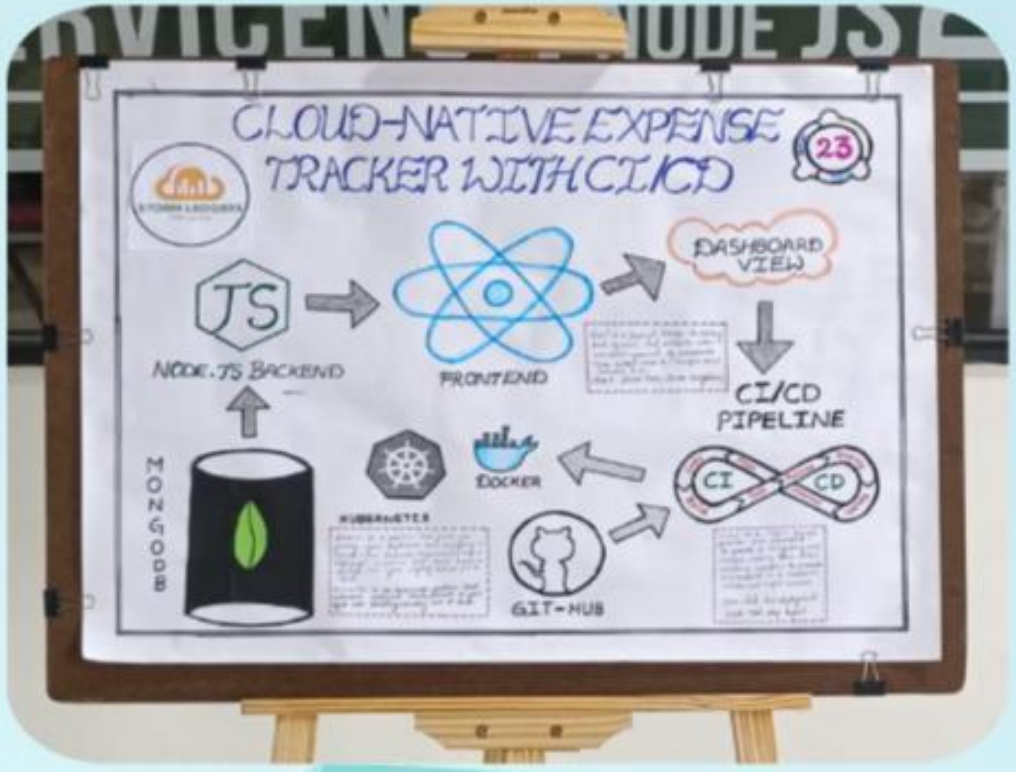
Code changes are versioned with Git, built into Docker images, and orchestrated with Kubernetes for automated and continuous cloud deployment.

03

Version Control

Integrate Git for tracking changes, enabling seamless collaboration and rollbacks if needed.

Technology Stack and Rationale



Security

Implement secure user authentication with custom registration and login, along with role-based access controls.

Scalability

Design horizontally scalable services to handle high traffic using cloud-native tools like Kubernetes.

Reliability

Ensures high availability and fault tolerance through automated container health checks, restarts, and failover handling.

Key Features

Scalability

Ensure seamless scaling capabilities to accommodate increasing user demands and data growth.

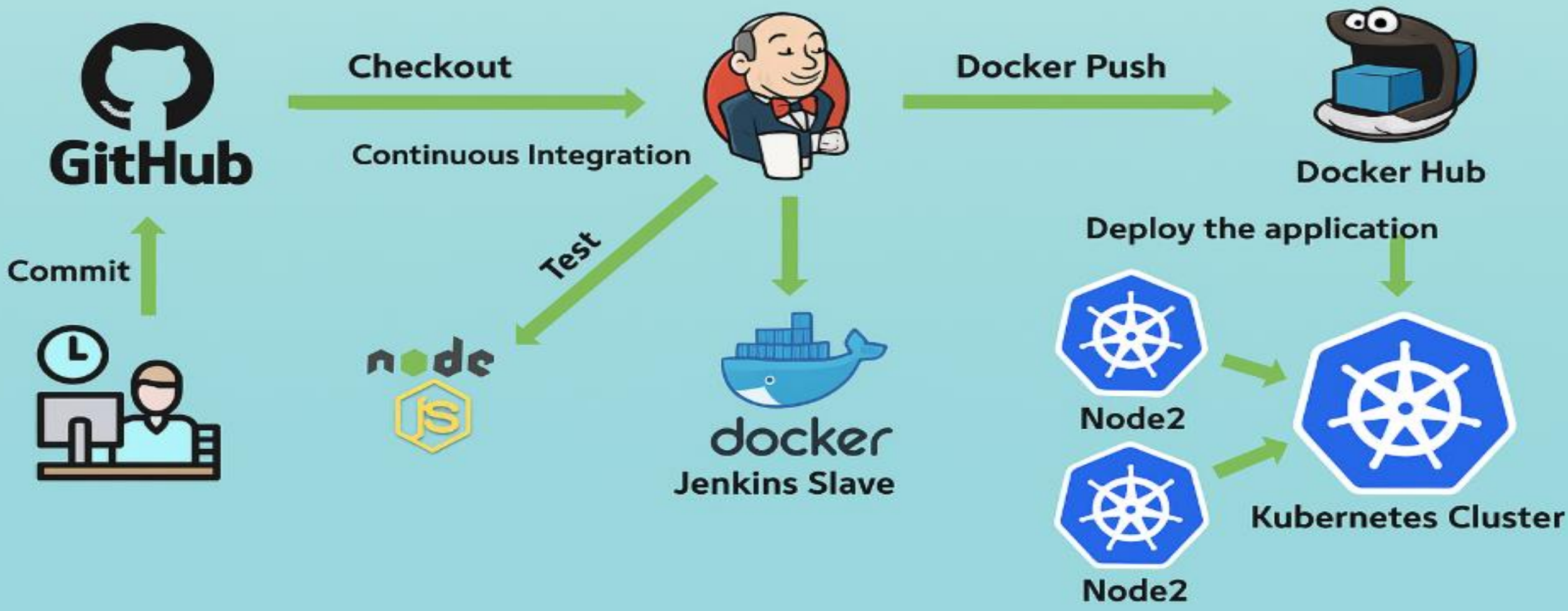
Automation

Integrate robust Continuous Integration/Continuous Deployment (CI/CD) pipelines to automate testing and deployments.

Cross-platform accessibility

Enable users to track expenses from multiple devices and platforms.

CI/CD PIPELINE FOR KUBERNETES DEPLOYMENT USING JENKINS



THANK YOU