ADOPTING A MULTI-CLOUD STRATEGY WITH DOCKER AND KUBERNETES

PHASE 2 - SOLUTION ARCHITECTURE

College Name: K S Institute of Technology

Group Members:

- **Name: Challa Deepika**

  CAN ID Number: CAN-33689518

- **Name: Vidheesha T M**

  CAN ID Number: CAN-33689420

- **Name: Arun A**

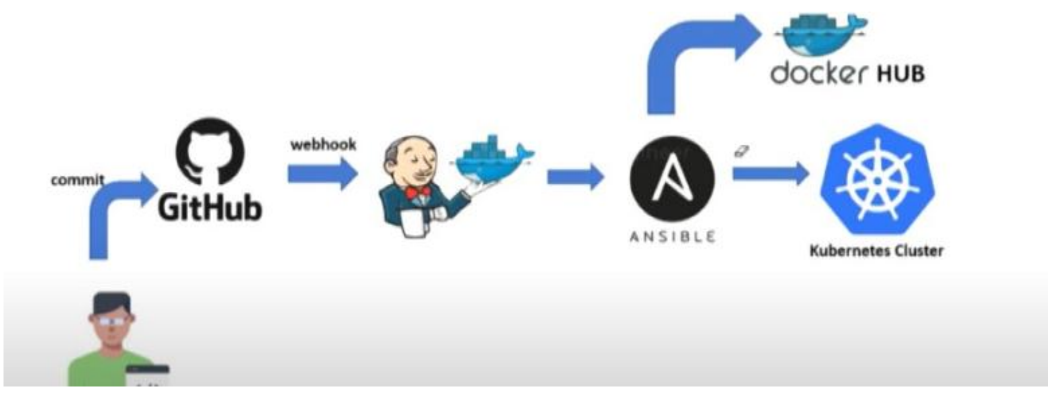
  CAN ID Number: CAN-33672150

- **Name: Chandrasekar A**

  CAN ID Number: CAN-33671259

SOLUTION ARCHITECTURE

To enable a scalable, resilient, and portable application, we will adopt a multi-cloud strategy using Docker, Kubernetes, and Ansible. This approach mitigates vendor lock-in while leveraging the strengths of different cloud platforms such as IBM Cloud, AWS, and Azure. The deployment process involves version control, CI/CD automation, container orchestration, and security measures to ensure seamless operations across multiple clouds.



PROJECT STRUCTURE

We will create a structured project directory to facilitate deployment across multiple clouds.

Project Directory Structure:

```
multi-cloud-app/
├── public/
│   ├── css/style.css  # Frontend styles
│   ├── js/app.js      # Frontend scripts
│   └── index.html     # Main frontend page
├── server/
│   ├── controllers/productController.js  # Business logic
│   ├── models/productModel.js          # Database schema
│   ├── routes/productRoutes.js         # API routes
│   └── server.js                       # Backend server
├── Dockerfile           # Containerization
├── docker-compose.yml     # Local container orchestration
├── package.json          # Node.js dependencies
├── Deployment.yml        # Kubernetes deployment
├── Service.yml           # Kubernetes service
├── Ansible.yml           # Ansible automation playbook
└── README.md             # Documentation
```

VERSION CONTROL SETUP

We will use GitHub for version control and CI/CD automation.

```
git init
echo node_modules/ > .gitignore
echo .env >> .gitignore
git add .
git commit -m "Initial commit"
git remote add origin <repository_url>
git push -u origin master
```

CI/CD PIPELINE DESIGN AND IMPLEMENTATION

To automate multi-cloud deployments, we will implement a Jenkins-based CI/CD pipeline integrated with Ansible.

Jenkins Pipeline Overview:

1. **Checkout Stage: Pull the latest code from GitHub.**
2. **Build Stage: Create Docker images.**
3. **Push Stage: Store images in cloud registries.**
4. **Deploy Stage: Deploy to Kubernetes using Ansible.**

Ansible Playbook for Deployment:

```
- hosts: all
  become: true
  tasks:
     - name: create new deployment
       command: kubectl apply -f /home/ubuntu/Deployment.yml
     - name: create new service
       command: kubectl apply -f /home/ubuntu/Service.yml
```

FUTURE PLANS

1. **Multi-Cloud Cluster Management: Implement KubeSphere for centralized management.**
2. **Security Enhancements: Integrate Kubernetes RBAC and network policies.**
3. **Performance Monitoring: Use Grafana and Prometheus for real-time tracking.**
4. **Advanced CI/CD Features: Implement GitHub Actions for automated testing and deployment.**