ADOPTING A MULTI-CLOUD STRATEGY WITH DOCKER AND KUBERNETES

PHASE 1 - 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

PROBLEM ANALYSIS

To enable a scalable, resilient, and portable cloud-native application, we will adopt a multi-cloud strategy using Docker and Kubernetes. This approach mitigates vendor lock-in while leveraging the strengths of different cloud platforms such as AWS, Azure, and Google Cloud. The deployment process will involve containerization, workload orchestration, automation, and security practices to streamline operations across multiple clouds.

Additionally, we will incorporate Ansible to automate Kubernetes deployments, ensuring faster and error-free updates to applications.

KEY CHALLENGES IDENTIFIED

1. **Complex Workload Management – Distributing and managing applications across multiple cloud providers.**
2. **Inconsistent Deployment Pipelines – Varying tools and processes leading to inefficiencies.**
3. **Portability Issues – Ensuring seamless execution across diverse cloud environments.**
4. **Security & Compliance – Maintaining consistent security policies and compliance standards.**
5. **Cost Optimization – Balancing resources effectively to avoid unnecessary expenses.**

SOLUTION APPROACH

Integration Needs

- **Utilize Docker for application containerization.**
- **Deploy Kubernetes for managing workloads across multiple cloud environments.**
- **Automate deployments using Ansible.**

Multi-Cloud Challenges

- **Ensuring consistent deployment and scaling policies.**
- **Managing networking configurations and reducing latency between cloud platforms.**

Automation Goals

- **Automate deployment and scaling using Jenkins and Ansible.**
- **Implement continuous monitoring for performance optimization.**

PROJECT STRUCTURE

Project Directory Setup:

```
multi-cloud-app/
├── public/
│   ├── css/style.css  # Frontend styles
│   ├── js/app.js      # Frontend scripts
│   └── index.html     # Main frontend page
├── server/
│   ├── controllers/mainController.js  # Business logic
│   ├── models/userModel.js          # Database schema
│   ├── routes/mainRoutes.js         # API routes
│   └── app.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 streamline deployments across multi-cloud environments, we will set up 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.**