

ADOPTING A MULTI-CLOUD STRATEGY WITH DOCKER AND KUBERNETES**PHASE 3 - 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 DEVELOPMENT

Setting up Multi-Cloud Environment and Configuring Necessary Tools

Step 1: Set Up Accounts on Multiple Cloud Providers

1. Create accounts on IBM Cloud, AWS, and Azure.
2. Ensure billing accounts are configured to access services across these platforms.

Step 2: Install Required Tools Locally

1. Install Minikube - Enable local Kubernetes testing.
2. Install kubectl - Set up Kubernetes CLI for cluster management.
3. Install Docker - Manage container images and deployments.

Step 3: Configure Multi-Cloud Container Registries

1. Set up registries on IBM Cloud, AWS, and Azure:

```
ibmcloud cr namespace-add <namespace_name>
```

```
aws ecr create-repository --repository-name <repository_name>
```

```
az acr create --name <registry_name> --resource-group <resource_group> --sku Basic
```

2. Enable security policies and vulnerability scanning for each registry.

IMPLEMENTATION DETAILS

Containerization Using Docker

Dockerfile (Backend Service):

FROM node:16-alpine

WORKDIR /app

COPY . .

RUN npm install

EXPOSE 5000

CMD ["node", "server.js"]

Building and Pushing Docker Images

docker build -t backend-app:latest .

docker tag backend-app:latest <REGISTRY_URL>/<namespace>/backend-app:latest

docker push <REGISTRY_URL>/<namespace>/backend-app:latest

Deploying to Kubernetes

Deployment YAML (backend-deployment.yml)

apiVersion: apps/v1

kind: Deployment

metadata:

name: backend-deployment

spec:

replicas: 3

selector:

matchLabels:

app: backend

template:

metadata:

labels:

app: backend

spec:

containers:

- name: backend

image: <REGISTRY_URL>/<namespace>/backend-app:latest

ports:

- containerPort: 5000

Service YAML (backend-service.yml)

apiVersion: v1

kind: Service

metadata:

name: backend-service

spec:

selector:

app: backend

ports:

- protocol: TCP

port: 80

targetPort: 5000

type: LoadBalancer

TESTING THE SOLUTION

Step 1: Deploy Applications to Multi-Cloud Kubernetes Clusters

- **Use Minikube for local testing or IBM Kubernetes Service, AWS EKS, and Azure AKS for production.**
- **Start Minikube:**

minikube start

- **Deploy Kubernetes manifests:**

kubectl apply -f backend-deployment.yml

kubectl apply -f backend-service.yml

Step 2: Verify Deployments

kubectl get pods

kubectl get svc

- **Access the application via public IPs or NodePort configurations.**

Step 3: Automate Multi-Cloud CI/CD Pipelines

- **Use Jenkins, GitHub Actions, or IBM Continuous Delivery.**
- **Implement an Ansible-based Kubernetes deployment:**

- hosts: all

become: true

tasks:

- **name:** create new deployment

command: `kubectl apply -f /home/ubuntu/backend-deployment.yml`

- **name:** create new service

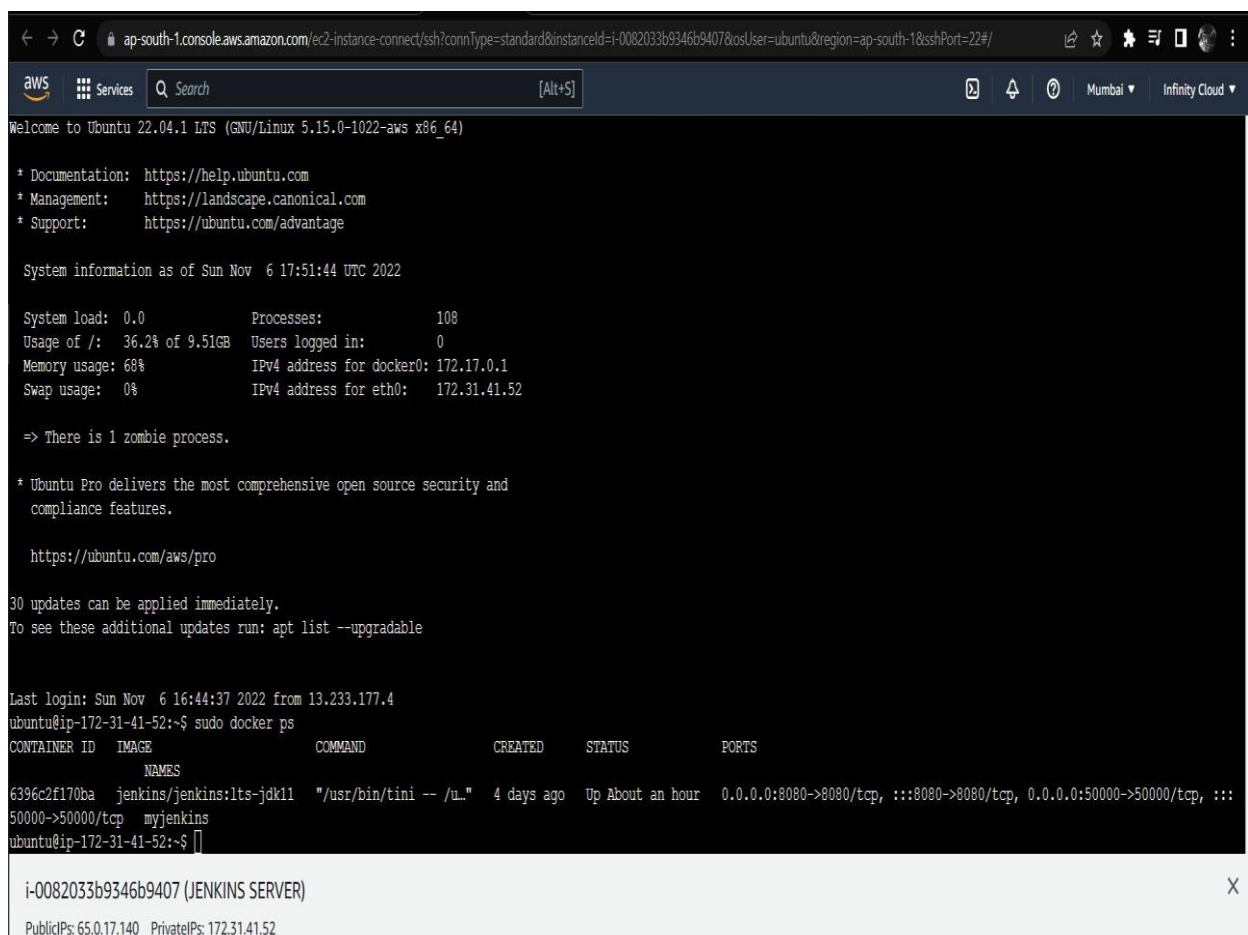
command: `kubectl apply -f /home/ubuntu/backend-service.yml`

Step 4: Conduct Performance Testing

- Use Apache JMeter or Postman for load testing.
- Monitor using Grafana or Azure Monitor.

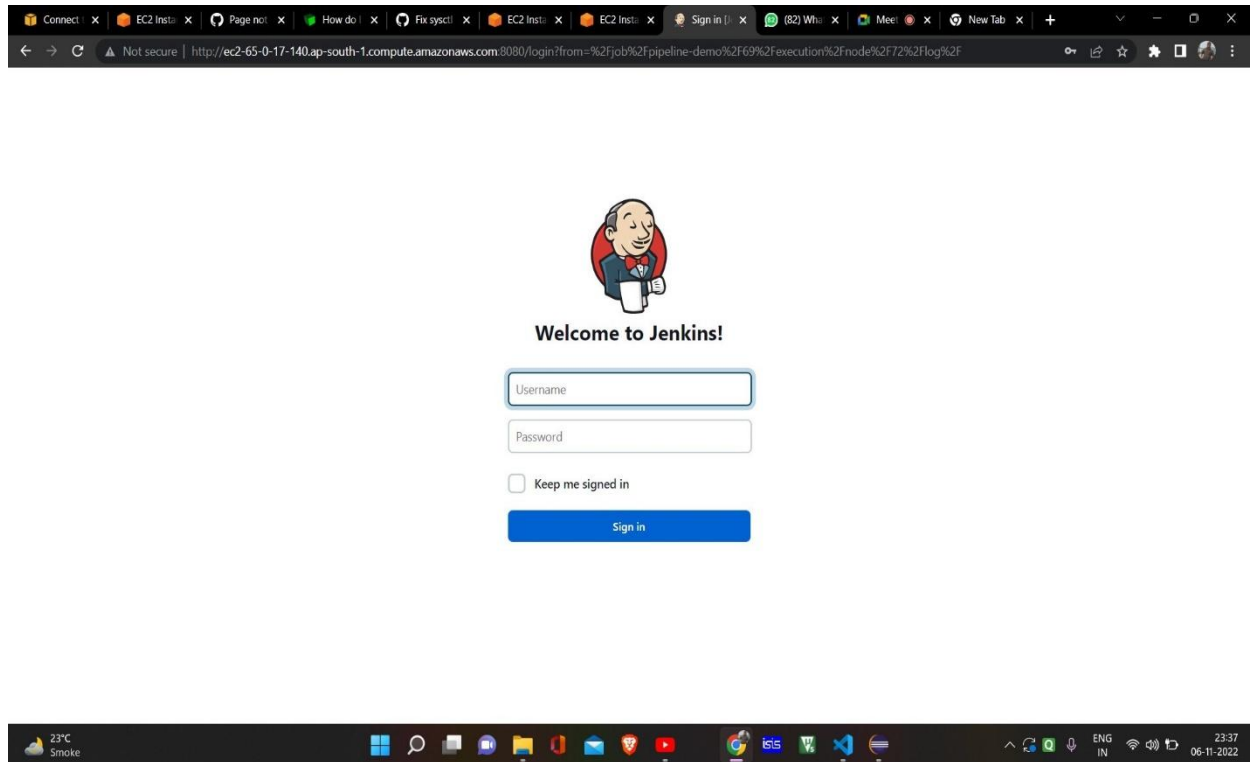
FUTURE IMPROVEMENTS

1. **Enable Auto-Scaling: Implement Kubernetes HPA.**
2. **Leverage Service Meshes: Deploy Istio for advanced networking.**
3. **Integrate Security Scanning: Automate real-time security analysis within CI/CD pipelines.**



```
ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0082033b9346b9407&osUser=ubuntu&region=ap-south-1&sshPort=22#/  
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1022-aws x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
System information as of Sun Nov  6 17:51:44 UTC 2022  
  
System load:  0.0               Processes:            108  
Usage of /:   36.2% of 9.51GB   Users logged in:     0  
Memory usage: 68%              IPv4 address for docker0: 172.17.0.1  
Swap usage:   0%               IPv4 address for eth0:   172.31.41.52  
  
=> There is 1 zombie process.  
  
* Ubuntu Pro delivers the most comprehensive open source security and  
  compliance features.  
  
https://ubuntu.com/aws/pro  
  
30 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
Last login: Sun Nov  6 16:44:37 2022 from 13.233.177.4  
ubuntu@ip-172-31-41-52:~$ sudo docker ps  
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS  
6396c2f170ba   jenkins/jenkins:lts-jdk11          "/usr/bin/tini -- /u..." 4 days ago    Up About an hour 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:50000->50000/tcp, :::50000->50000/tcp  
myjenkins  
ubuntu@ip-172-31-41-52:~$  
  
i-0082033b9346b9407 (JENKINS SERVER)  
PublicIPs: 65.0.17.140 PrivateIPs: 172.31.41.52
```

PHASE 3



DEVOPS ENGINEER