PHASE 2

DevOps Engineer
Establishing CI/CD Pipeline for Automated Deployment

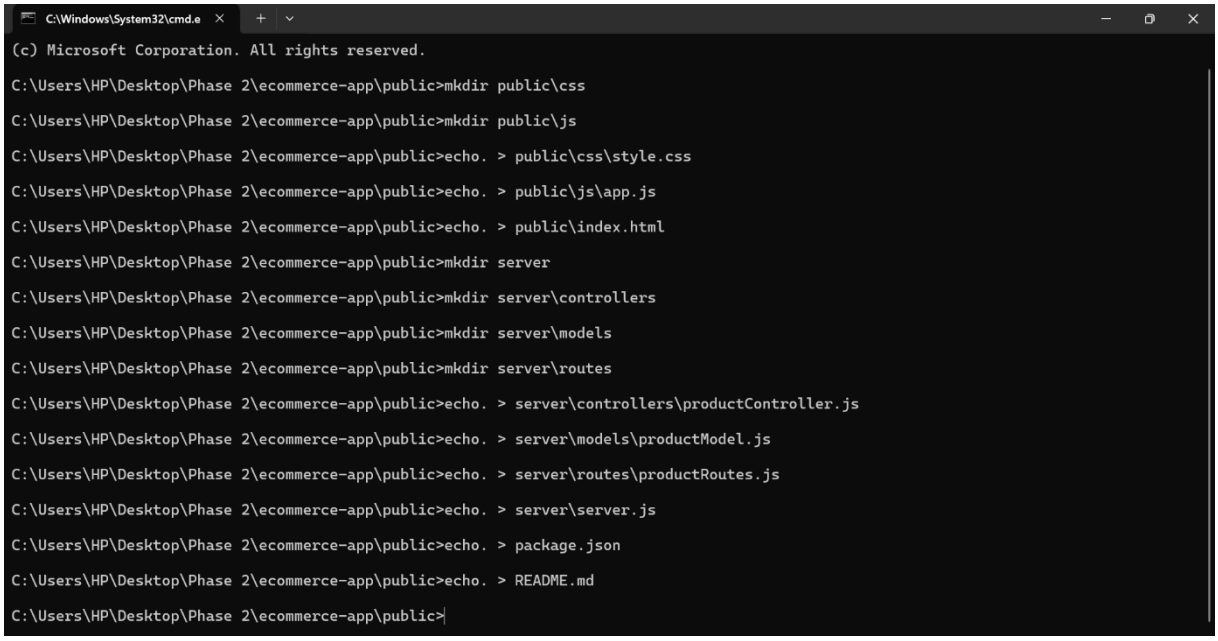
Problem Analysis

College Name: SKSVMACET, Lakshmeshwar
Group Members:

- **Name**: Suman R Devagiri
  CAN ID Number: CAN_33892921
- **Name**: Sunita Kumbar
  CAN ID Number: CAN_33891602
- **Name**: Kavya Banni
  CAN ID Number: CAN_33891836
- **Name**: Soujanya Parananti
  CAN ID Number: CAN_33893098

SOLUTION ARCHITECTURE

To automate the deployment of containerized applications, we will design a CI/CD pipeline leveraging version control, automation tools, and deployment orchestration on Kubernetes. The pipeline ensures streamlined builds, testing, and deployments, enhancing efficiency and reliability.



```
C:\Users\HP>echo node_modules/ > .gitignore

C:\Users\HP>git init
Reinitialized existing Git repository in C:/Users/HP/.git/

C:\Users\HP>echo node_modules/ > .gitignore

C:\Users\HP>echo .env > .gitignore
```

```
C:\Users\HP\Desktop\Phase 2>git add .

C:\Users\HP\Desktop\Phase 2>git commit -m "Initial commit of ecommerce-app structure"
[main (root-commit) 3757021] Initial commit of ecommerce-app structure
 9 files changed, 9 insertions(+)
 create mode 100644 Desktop/Phase 2/ecommerce-app/public/README.md
 create mode 100644 Desktop/Phase 2/ecommerce-app/public/package.json
 create mode 100644 Desktop/Phase 2/ecommerce-app/public/public/css/style.css
 create mode 100644 Desktop/Phase 2/ecommerce-app/public/public/index.html
 create mode 100644 Desktop/Phase 2/ecommerce-app/public/public/js/app.js
 create mode 100644 Desktop/Phase 2/ecommerce-app/public/server/controllers/productController.js
 create mode 100644 Desktop/Phase 2/ecommerce-app/public/server/models/productModel.js
 create mode 100644 Desktop/Phase 2/ecommerce-app/public/server/routes/productRoutes.js
 create mode 100644 Desktop/Phase 2/ecommerce-app/public/server/server.js

C:\Users\HP\Desktop\Phase 2>git remote add origin https://github.com/sumandevagiri/IBM_Phase-2.git
```

```
C:\Users\HP\Desktop\Phase 2>git pull origin main
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 870 bytes | 39.00 KiB/s, done.
From https://github.com/sumandevagiri/IBM_Phase-2
 * branch            main       -> FETCH_HEAD
 * [new branch]      main       -> origin/main
fatal: refusing to merge unrelated histories
```

```
C:\Users\HP\Desktop\Phase 2>git push -u origin main
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (14/14), 1.06 KiB | 180.00 KiB/s, done.
Total 14 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/sumandevagiri/IBM_Phase-2.git
   5035406..d747014  main -> main
branch 'main' set up to track 'origin/main'.
```

```
C:\Users\HP\Desktop\Phase 2>git push -u origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/sumandevagiri/IBM_Phase-2/pull/new/master
remote:
To https://github.com/sumandevagiri/IBM_Phase-2.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

LastPass Gmail YouTube Maps Booking.com Express VPN McAfee Security LastPass password... Other favorites

Dashboard > ibm > #1

- Changes
- Console Output
- Edit Build Information
- Delete build '#1'
- Timings
- Git Build Data
- Pipeline Overview
- Pipeline Console
- Restart from Stage
- Replay
- Pipeline Steps
- Workspaces

```
Started by user Suman R Devagiri
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\ibm
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Checkout)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\ibm\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/sumandevagiri/jenkines.git # timeout=10
Fetching upstream changes from https://github.com/sumandevagiri/jenkines.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.47.0.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/sumandevagiri/jenkines.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10
Checking out Revision 4582970eb883392383283b2f9af1c3b6b5b457b0 (refs/remotes/origin/main)
 > git.exe config core.sparsecheckout # timeout=10
```

Dashboard > ibm > #1

```
[Pipeline] { (Test)
[Pipeline] echo
Running tests...
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
Deploying the application...
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
Pipeline succeeded!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Jenkins | Search (CTRL+K) | 1 | Suman R Devagiri | log out

Dashboard > ibm > Stages

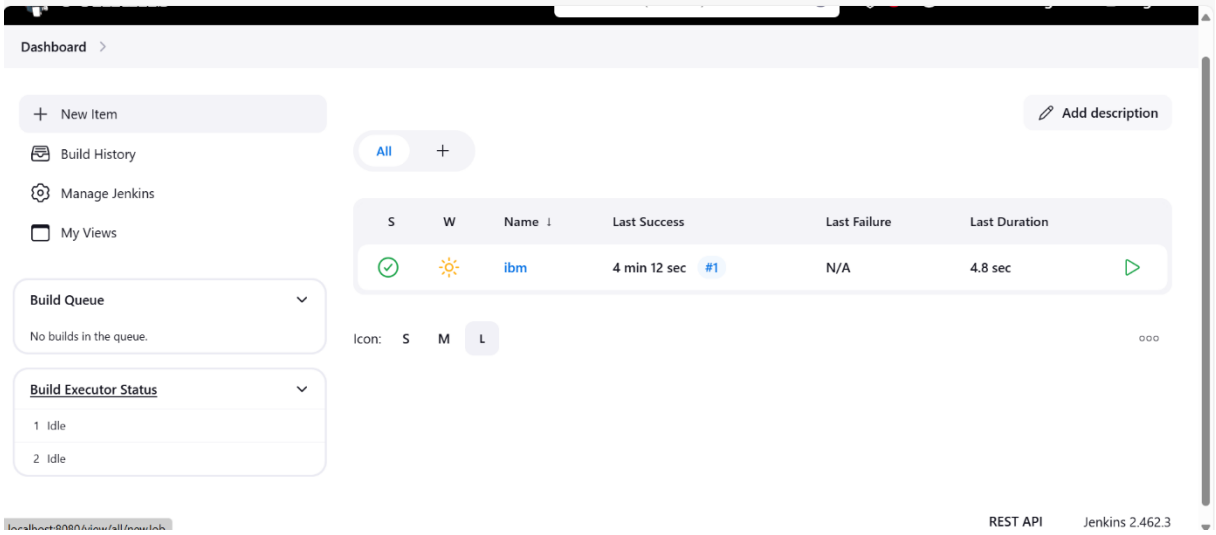
Build ibm

Build | Configure

id pipeline

#1 Start ✓ Checkout ✓ Build ✓ Test ✓ Deploy ✓ *Post Actions* ✓ End

Jenkins 2.462.3




```
C:\Users\HP>ibmcloud --version
ibmcloud 2.31.0 (6b1eddc-2024-12-05T17:30:20+00:00)
Copyright IBM Corp. 2014, 2024
```

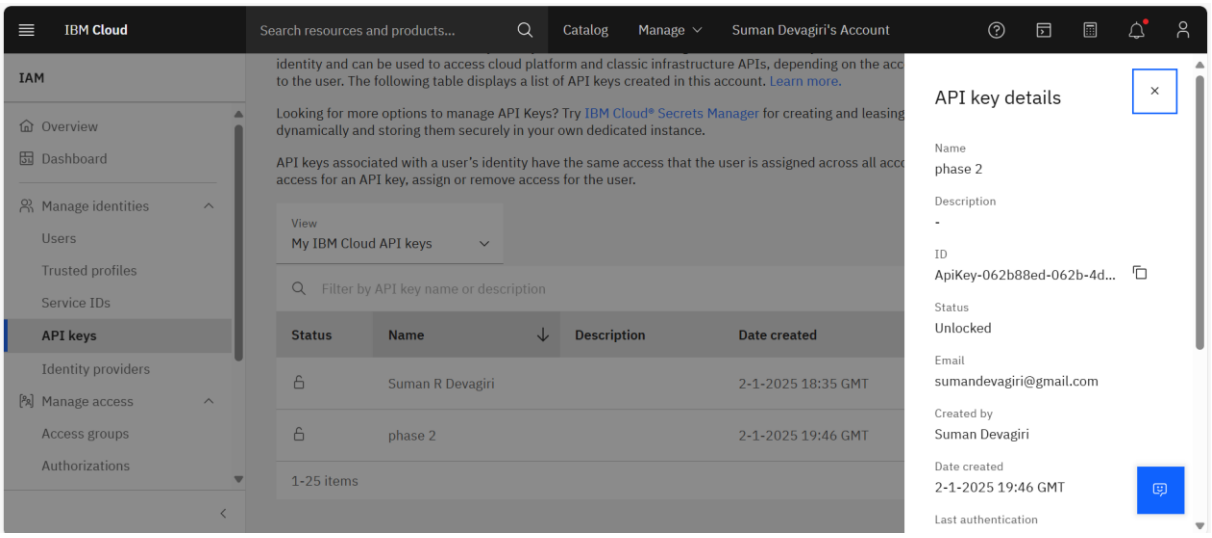
```
Command Prompt
C:\Users\HP>ibmcloud login --apikey Ig0WLlx9SRvb6fufJHI6cI6NGapFWe5-QMFQ20qaZVjW
API endpoint: https://cloud.ibm.com
Authenticating...
OK

Targeted account Suman Devagiri's Account (8d64708c57d1473da56ef4cabebe804c)


Select a region (or press enter to skip):
1. au-syd
2. in-che
3. jp-osa
4. jp-tok
5. eu-de
6. eu-es
7. eu-gb
8. ca-tor
9. us-south
10. us-east
11. br-sao
Enter a number>


API endpoint:     https://cloud.ibm.com
Region:
User:             sumandevagiri@gmail.com
Account:          Suman Devagiri's Account (8d64708c57d1473da56ef4cabebe804c)
Resource group:   No resource group targeted, use 'ibmcloud target -g RESOURCE_GROUP'

C:\Users\HP>
```



```
C:\Users\HP>ibmcloud plugin install container-registry
Looking up 'container-registry' from repository 'IBM Cloud'...
Plug-in 'container-registry[cr] 1.3.12' found in repository 'IBM Cloud'
Attempting to download the binary file...
 12.25 MiB / 12.25 MiB [==========================================================================================] 100.00% 28s
12841472 bytes downloaded
Installing binary...
OK
Plug-in 'container-registry 1.3.12' was successfully installed into C:\Users\HP\.bluemix\plugins\container-registry. Use 'ibmcloud pl
ugin show container-registry' to show its details.

C:\Users\HP>ibmcloud plugin list
Listing installed plug-ins...

Error: Unable to fetch plug-ins from repository 'IBM Cloud':
Get "https://download.clis.cloud.ibm.com/ibm-cloud-cli-plugin-metadata/bluemix-repo-index.yml": dial tcp: lookup download.clis.cloud.
ibm.com: no such host

Plugin Name              Version   Status   Private endpoints supported
container-registry[cr]   1.3.12             true
```

```
C:\Users\HP>ibmcloud plugin list
Listing installed plug-ins...

Error: Unable to fetch plug-ins from repository 'IBM Cloud':
Get "https://download.clis.cloud.ibm.com/ibm-cloud-cli-plugin-metadata/bluemix-repo-index.yml": dial tcp: lookup download.clis.cloud.
ibm.com: no such host

Plugin Name              Version   Status   Private endpoints supported
container-registry[cr]   1.3.12             true
```

```
C:\Users\HP>ibmcloud cr login
Logging 'docker' in to 'icr.io'...
Logged in to 'icr.io'.

OK

C:\Users\HP>
```

Jenkins File:

```
pipeline {
    agent any
    environment {
        DOCKER_IMAGE = 'my-app'
        REGISTRY_URL = '<IBM_Container_Registry_URL>' // Replace with the actual IBM
Cloud Container Registry URL
        CLUSTER_NAME = '<CLUSTER_NAME>' // Replace with your Kubernetes cluster
name
    }
    stages {
        stage('Checkout') {
            steps {
                git 'https://github.com/sumandevagiri/IBM_Phase-2' // Replace <username> with
the actual GitHub username
            }
        }
        stage('Build Docker Image') {
            steps {
                script {
                    sh 'docker build -t $REGISTRY_URL/$DOCKER_IMAGE .'
                }
            }
        }
        stage('Push Docker Image to IBM Cloud Container Registry') {
            steps {
                script {
                    sh 'docker push $REGISTRY_URL/$DOCKER_IMAGE'
                }
```

```
            }
        }
        stage('Deploy to Kubernetes') {
            steps {
                script {
                    sh '''
                        ibmcloud login --apikey Ig0WLlx9SRvb6fufJHI6cI6NGapFWe5-
QMFQ20qaZVjW -r <REGION> -g <RESOURCE_GROUP> // Replace placeholders with
your credentials
                        ibmcloud ks cluster config --cluster $CLUSTER_NAME
                        kubectl apply -f k8s/deployment.yaml
                    '''
                }
            }
        }
    }
    post {
        success {
            echo 'Pipeline executed successfully.'
        }
        failure {
            echo 'Pipeline failed. Please check the logs.'
        }
    }
}
```

FUTURE PLAN:

1. Container Image Management with IBM Cloud Container Registry: Utilize IBM Cloud Container Registry for storing and managing Docker images. This will provide a secure and centralized storage for the container images, enabling streamlined deployments.

2. Docker Image Build and Deployment: Build the Docker image for the frontend and backend services and push them to IBM Cloud Container Registry for storage and future deployments.

3. Kubernetes Cluster Setup and Deployment: Deploy the Dockerized application on a Kubernetes cluster, using Minikube for local development and testing. This will allow us to simulate a production-like environment, ensuring scalability and resilience.

4. Security with OpenSSL: Implement OpenSSL for secure image management and encryption. This will include vulnerability scanning and signing the Docker images to ensure that only trusted versions are deployed, improving the security of the deployment pipeline.

5. CI/CD Pipeline Integration: Automate the build, test, and deployment process by integrating IBM Cloud Continuous Delivery, Jenkins, or GitHub Actions with the Kubernetes deployment pipeline, ensuring rapid and consistent updates.