A PROJECT REPORT

in fulfilment of the requirement of the 6th

Semester INDUSTRIAL TRAINING

BACHELOR OF ENGINEERING (BE)

in

Computer Science & Engineering

Under the guidance of

Narender Kumar Jain

Scientist-E

NATIONAL

INFORMATICS CENTER.

(NIC)

Submitted by

ROHINI SINGH

GCS - 2231054

SANT LONGOWAL INSTITUTE OF ENGINEERING AND TECHNOLOGY

Sangrur, Punjab - 148106

(Deemed-to-be-University, Under MOE)

July, 2024

ACKNOWLEDGEMENT

Firstly, I impart categorical gratitude to the almighty god for their divine blessings and for giving me the strength to complete this internship at the NATIONAL INFORMATICS CENTER from the 3rd of June to the 31th of July with success.

My training would not have been possible without the contribution and collaboration of others. My sincere gratitude:
Sh. Narender Kumar Jain is the Scientist-E ,1 want to thank him for allowing me to follow the training under his guidance. Deep information and the keen interest of my supervisor in the website application design and development field influenced me to hold out this spot. His endless patience, studious steering, continual encouragement, constant and energetic oversight, constructive criticism, valuable recommendation, reading several inferior drafts and correcting them at the least bit stage have made it attainable to finish this spot.

I wish to impart my all-course mates in SLIET University, who took half during this discussion while finishing the course work. Finally, I must acknowledge with due respect and love, the constant support and encouragement of my family and one of my close friends throughout this training.

ABSTRACT

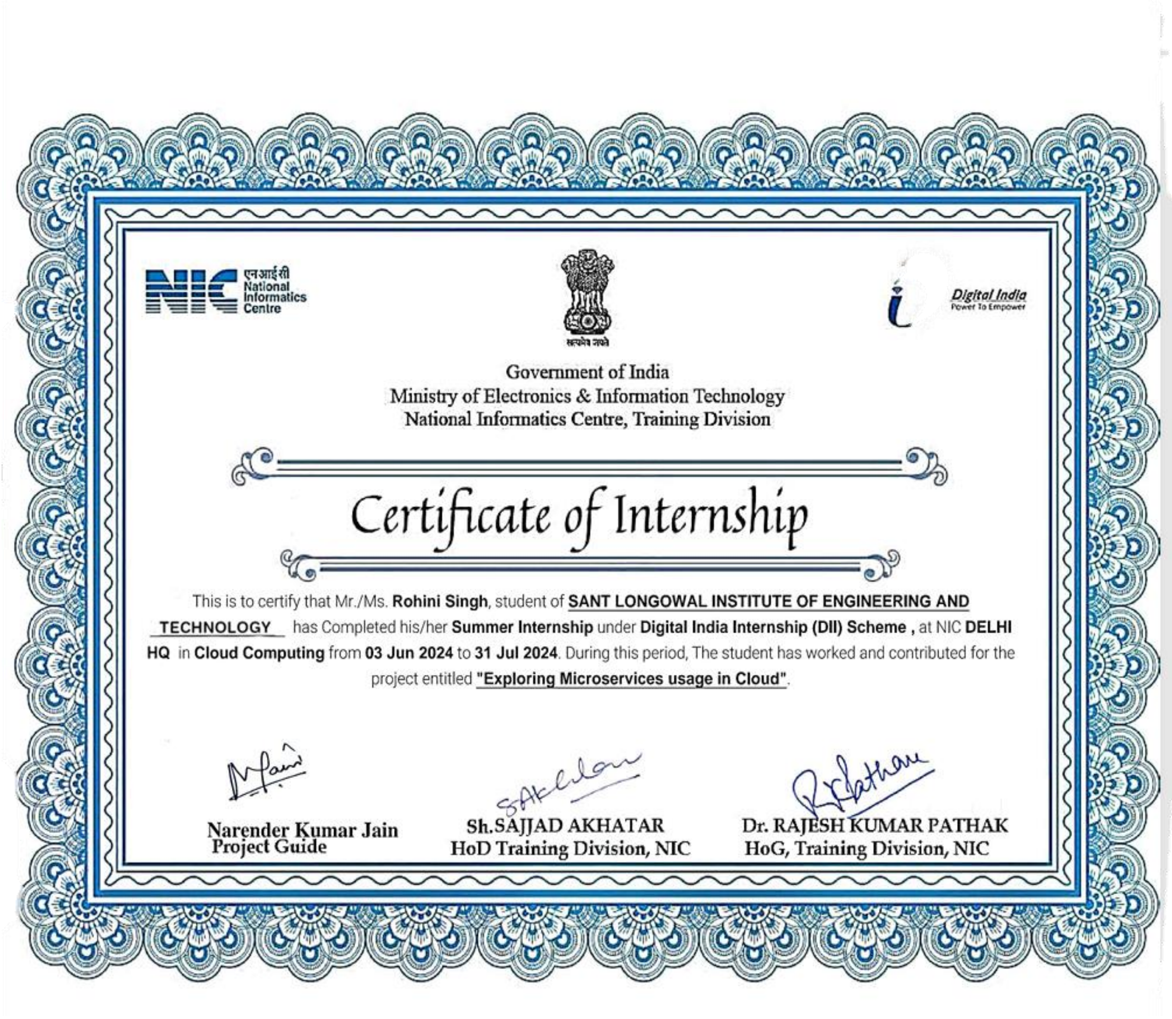
This report provides an overview of the key learnings and experiences gained during my internship, focusing on cloud computing, Linux, OpenStack, monitoring tools, Docker, and Kubernetes. The internship offered a comprehensive exploration of these technologies through both theoretical study and practical application. Key areas covered include:

- **Cloud Computing**: Understanding of various cloud service models (IaaS, PaaS, SaaS) and deployment types (public, private, hybrid), along with their respective benefits and challenges.
- **Linux Basics**: Proficiency in Linux file system structure, essential commands, and the vi editor for file editing.
- **OpenStack**: Knowledge of core components and their integration for managing cloud infrastructure.
- **Monitoring Tools**: Skills in using Grafana for visualization, Prometheus for monitoring and alerting, and Loki for log aggregation.
- **Docker**: Experience in containerization, including developing a Dockerized application and using Docker Compose for multi-container setups.
- **Kubernetes**: Foundational understanding of Kubernetes concepts for container orchestration, demonstrated through a project on containerized application management.

The hands-on projects and training provided a robust foundation in cloud computing and DevOps practices, preparing me for advanced roles in IT and cloud infrastructure.

TABLE OF CONTENT

CERTIFICATE



एनआईसी National Informatics Centre

सत्यमेव जयते

Digital India Power To Empower

Government of India
Ministry of Electronics & Information Technology
National Informatics Centre, Training Division

Certificate of Internship

This is to certify that Mr./Ms. **Rohini Singh**, student of **SANT LONGOWAL INSTITUTE OF ENGINEERING AND TECHNOLOGY** has Completed his/her **Summer Internship** under **Digital India Internship (DII) Scheme ,** at NIC **DELHI HQ** in **Cloud Computing** from **03 Jun 2024** to **31 Jul 2024**. During this period, The student has worked and contributed for the project entitled **"Exploring Microservices usage in Cloud"**.

[illegible] [illegible] [illegible]

Narender Kumar Jain	Sh.SAJJAD AKHATAR	Dr. RAJESH KUMAR PATHAK
Project Guide	HoD Training Division, NIC	HoG, Training Division, NIC

ABOUT THE COMPANY

National Informatics Centre (NIC) under the Ministry of Electronics and Information Technology (MeitY) is the technology partner of the Government of India. It was established in 1976 with an objective to provide technology-driven solutions to Central and State Governments in various aspects of development. NIC has been instrumental in adopting and providing Information and Communication Technology (ICT) and eGovernance support to Central Government.

Its state-of-art IT infrastructure includes Multi-Gigabit PAN India Network NICNET, National Knowledge Network, National Data Centers, National Cloud, Video Conferencing, Email and Messaging Services, Command and Control Centre, Multi-layered GIS based Platform, Domain Registration and Webcast.

NIC has also developed several digital platforms for the socio-economic development of the country with 'One-Nation One-Platform' initiative to empower citizens digitally. Its services have created a perfect interaction of the Government with citizens, Government employees and businesses. With an objective of focused study of new technology, explore and experiment their use in governance, NIC has set-up Centre of Excellence (CoE) in Data Analytics, Artificial Intelligence, Blockchain and Application Security.

एनआईसी
National
Informatics
Centre

Services provided by NIC

- NICNET
- Data Centre
- National Cloud
- Centralised Aadhaar Vault
- Security
- Video Conferencing
- Government Local Area Networks (LANs)

INTRODUCTION

The internship provided an in-depth exploration of cloud computing and DevOps tools, essential for modern IT infrastructure and development. This training covered a broad spectrum, from foundational concepts in cloud computing to advanced container orchestration with Kubernetes. The primary goal was to equip me with a comprehensive understanding of these technologies and their practical applications.

Problem Statement

In today's rapidly evolving IT landscape, managing and optimizing infrastructure can be complex and challenging. Traditional approaches often struggle with scalability, efficiency, and integration, necessitating a modernized approach to infrastructure management and deployment. The internship aimed to address these challenges by introducing key technologies and methodologies that streamline and enhance IT operations.

Objectives

The main objectives of the internship were to:

1. **Understand Cloud Computing**: Gain knowledge of cloud service models (IaaS, PaaS, SaaS) and deployment strategies (public, private, hybrid) to appreciate their benefits and challenges.
2. **Master Linux Basics**: Develop proficiency in Linux fundamentals, including file system structure, essential commands, and the vi editor, to manage and navigate Linux environments effectively.
3. **Explore OpenStack**: Learn about OpenStack's core components and their integration to manage cloud infrastructure, enabling efficient resource management.
4. **Utilize Monitoring Tools**: Acquire skills in Grafana for data visualization, Prometheus for monitoring and alerting, and Loki for log aggregation to enhance system observability and performance monitoring.
5. **Implement Docker**: Gain hands-on experience with Docker to understand containerization concepts, develop Dockerized applications, and manage containers and images.
6. **Understand Kubernetes**: Develop foundational knowledge in Kubernetes for container orchestration, including managing containerized applications and deploying scalable solutions.

This report outlines the key areas covered during the internship, the practical projects completed, and the skills acquired, showcasing how these experiences contribute to modern IT and cloud infrastructure management.

INTERNSHIP OUTCOME

The internship offered a comprehensive exploration of cloud computing and DevOps tools, crucial for modern IT infrastructure and development. This hands-on experience covered a range of technologies and methodologies, leading to several key outcomes:

1. **Cloud Computing Foundations**:
   - Gained a solid understanding of cloud service models (IaaS, PaaS, SaaS) and deployment strategies (public, private, hybrid). This foundational knowledge facilitated a clear grasp of how cloud environments operate and their impact on IT infrastructure.
2. **Proficiency in Linux**:
   - Developed practical skills in Linux, including familiarity with essential commands and the vi editor. This knowledge enhanced my ability to navigate and manage Linux-based systems effectively.
3. **Hands-On Experience with OpenStack**:
   - Acquired a foundational understanding of OpenStack, including its core components and their integration. This experience provided insights into managing and orchestrating cloud infrastructure resources.
4. **Skills in Monitoring and Observability**:
   - Learned to use Grafana for data visualization, Prometheus for monitoring and alerting, and Loki for log aggregation. These tools equipped me with the ability to monitor system performance, visualize data trends, and manage logs effectively.
5. **Docker Containerization**:
   - Gained hands-on experience with Docker, including containerizing applications and managing Docker images. This practical experience enhanced my understanding of containerization and its benefits in application deployment.
6. **Fundamentals of Kubernetes**:
   - Acquired foundational knowledge in Kubernetes, including core concepts such as pods, services, and deployments. Successfully completed a project demonstrating the use of Kubernetes for container orchestration, highlighting my ability to manage and scale containerized applications.
7. **Practical Project Completion**:
   - Completed practical projects that integrated these technologies, reinforcing theoretical knowledge and demonstrating practical application. These projects showcased my ability to apply cloud and DevOps concepts in real-world scenarios.

In summary, the internship provided valuable hands-on experience and a deeper understanding of cloud computing and DevOps tools. The skills acquired and the projects completed during this period have significantly contributed to my readiness for advanced roles in IT infrastructure and development

Overview of Cloud:

Cloud computing delivers computing services over the internet, including storage, processing power, and applications. It offers scalable and flexible IT resources, reducing the need for physical infrastructure.

Key Learnings:

1. **Cloud Service Models:**
    - **IaaS (Infrastructure as a Service):** Provides virtualized computing resources over the internet, such as virtual machines and storage (e.g., AWS EC2, Microsoft Azure Virtual Machines).
    - **PaaS (Platform as a Service):** Offers a platform for developing, running, and managing applications without managing the underlying infrastructure (e.g., Google App Engine, Microsoft Azure App Services).
    - **SaaS (Software as a Service):** Delivers software applications over the internet on a subscription basis, with the provider managing the infrastructure (e.g., Google Workspace, Microsoft Office 365).
2. **Deployment Models:**
    - **Public Cloud:** Services are provided over the internet and shared among multiple users, offering cost-effectiveness and scalability (e.g., AWS, Google Cloud Platform).
    - **Private Cloud:** Dedicated to a single organization, offering enhanced security and control (e.g., VMware Private Cloud).
    - **Hybrid Cloud:** Combines public and private clouds, allowing for data and application sharing between them, providing flexibility (e.g., a mix of on-premises data centers and public cloud services).
3. **Benefits and Challenges:**
    - **Benefits:**
        - **Scalability:** On-demand resource scaling to meet varying workloads.
        - **Cost-Efficiency:** Pay-as-you-go model reduces upfront hardware costs.
        - **Flexibility:** Access to a wide range of tools and rapid deployment.
        - **Accessibility:** Services and data available from anywhere with internet access.
    - **Challenges:**
        - **Security:** Requires robust measures to protect data.
        - **Downtime:** Service outages can impact operations.
        - **Cost Management:** Requires careful monitoring to manage expenditures.
        - **Compliance:** Ensuring adherence to regulations and standards can be complex.

Overview of Linux:

Linux is a popular open-source operating system known for its stability and versatility. This section covers fundamental aspects of Linux, including its file system structure, basic commands, and the vi editor.

Key Learnings:

1. **File System Structure:**
    - **Directory Structure:** Understanding the hierarchical structure, including key directories such as /home, /etc, /var, and /usr.
    - **File Permissions:** Knowledge of file permissions (read, write, execute) and ownership, and how to modify them using commands like chmod and chown.
2. **Basic Commands:**
    - **ls:** Lists files and directories in a directory.
    - **cd:** Changes the current directory.
    - **cp:** Copies files and directories.
    - **mv:** Moves or renames files and directories.
    - **rm:** Removes files and directories.
3. **vi Editor:**
    - **Command Mode:** Used for navigation and executing commands within the vi editor.
    - **Insert Mode:** Allows for text entry and editing within a file.

These basics provide a strong foundation for navigating and managing Linux systems effectively.

<u>Overview of Virtualization:</u>

VirtualBox is a free and open-source virtualization tool that allows users to run multiple operating systems on a single physical machine. Virtualization enables the creation and management of virtual machines (VMs), providing a flexible and controlled environment for testing and development.

Virtualization:

Virtualization involves creating virtual instances of computing resources, such as servers, storage, or networks, on a single physical hardware platform. It allows multiple virtual environments to operate independently on a single physical system, optimizing resource utilization and providing isolation between different workloads.

Types of Virtualizations:

1. **Type 1 Hypervisor (Bare-Metal Hypervisor):**
    - **Overview:** A Type 1 hypervisor runs directly on the physical hardware of the host machine, without an underlying operating system. It manages and allocates resources to the virtual machines it hosts.
    - **Characteristics:**
        - **Performance:** Provides high performance and efficiency since it operates directly on the hardware.

        - **Security:** Offers better isolation and security as it does not rely on a host operating system.
        - **Examples:** VMware ESXi, Microsoft Hyper-V, Xen.
2. **Type 2 Hypervisor (Hosted Hypervisor):**
    - **Overview:** A Type 2 hypervisor runs on top of a conventional operating system. It relies on the host OS to manage hardware resources and provides virtual machines with access to these resources.
    - **Characteristics:**
        - **Ease of Use:** Easier to set up and use for desktop environments and development purposes.
        - **Performance:** May have slightly lower performance compared to Type 1 hypervisors due to the overhead of the host OS.
        - **Examples:** VirtualBox, VMware Workstation, Parallels Desktop.

Key Learnings:

1. **VirtualBox:**
    - **Virtualization:** VirtualBox is a Type 2 hypervisor that allows users to create and manage virtual machines on a host operating system.
    - **VM Configuration:** Configured a virtual machine with Ubuntu as the guest OS, managing resources such as disk space and memory.
2. **Creating a Virtual Machine on Ubuntu:**
    - **Installation:** Installed VirtualBox on an Ubuntu host system.
    - **VM Setup:** Created and configured a new Ubuntu VM for development purposes.
3. **First Project: Installing NGINX:**
    - **NGINX Installation:** Installed NGINX on the Ubuntu VM to serve web content.
    - **Configuration File Modification:** Edited the NGINX configuration file to set up a basic web server that displays a static HTML page.
    - **Static HTML Page:** Created and tested a static HTML page to be served by NGINX.

This project provided practical experience with virtualization using VirtualBox, and hands-on skills in server management and configuration with NGINX.

Overview OpenStack Basics:

OpenStack is an open-source platform for building and managing public and private clouds. It consists of various components that work together to deliver cloud services.

Key Components:

1. **Nova (Compute):** Manages and provisions virtual machines (instances) in the cloud, handling compute resources.

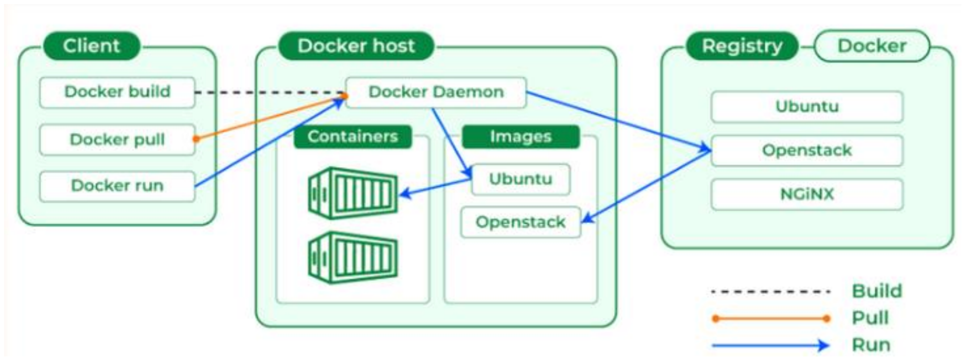
2. **Neutron (Networking):** Provides networking services, including network creation, management, and configuration.
3. **Cinder (Block Storage):** Manages block storage volumes, providing persistent storage for virtual machines.
4. **Swift (Object Storage):** Offers scalable and redundant object storage for large amounts of unstructured data.
5. **Glance (Image Service):** Manages and stores disk images and snapshots used for launching virtual machines.
6. **Keystone (Identity Service):** Provides authentication and authorization services, managing user identities and permissions.
7. **Horizon (Dashboard):** Web-based interface for managing OpenStack resources and services.
8. **Heat (Orchestration):** Automates resource deployment and scaling using templates, enabling infrastructure-as-code.
9. **Ceilometer (Telemetry):** Collects usage statistics and performance data for monitoring and billing.
10. **Aodh (Alarming):** Triggers alerts based on metrics collected by Ceilometer.
11. **Panko (Event Monitoring):** Monitors and stores event data related to OpenStack services.

<u>Overview of Docker:</u>

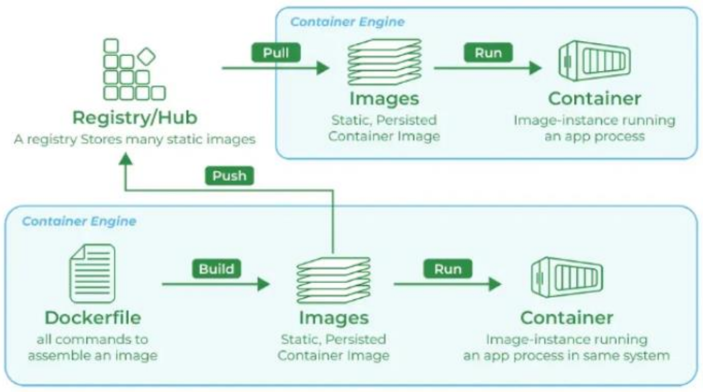
Docker is a platform that enables the development, shipping, and running of applications using containerization. Containers package an application and its dependencies into a single, portable unit. This section covers the basics of Docker and a project completed using Docker.

Key Learnings:

1. **Containerization:**
   - **Concept:** Containers encapsulate an application and its dependencies, ensuring consistency across different environments.
   - **Images:** Docker images are used to create containers, providing a snapshot of the application and its environment.



2. **Docker Commands:**
    - **docker run:** Used to start a new container from an image.
    - **docker build:** Creates a Docker image from a Dockerfile, defining the application's environment and setup.
    - **docker-compose:** Manages multi-container applications by defining services, networks, and volumes in a docker-compose.yml file.
3. **Project:**
    - **Dockerized Application:** Developed a project using Docker, showcasing containerization concepts by packaging an application into containers.
    - **Implementation:** Demonstrated the practical use of Docker commands and Docker Compose to manage and deploy the application efficiently.


Container Engine
Pull
Run
Registry/Hub
A registry Stores many static images
Images
Static, Persisted Container Image
Container
Image-instance running an app process
Push
Container Engine
Build
Run
Dockerfile
all commands to assemble an image
Images
Static, Persisted Container Image
Container
Image-instance running an app process in same system

This experience provided hands-on knowledge of Docker, illustrating the benefits of containerization for consistent and scalable application deployment.

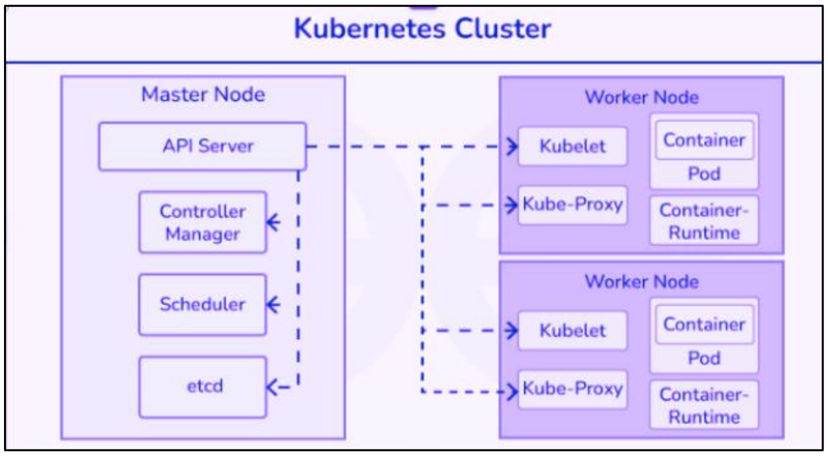
<u>Overview of Kubernetes Basics</u>

Kubernetes is an open-source platform designed for automating the deployment, scaling, and management of containerized applications. It provides a robust system for managing complex applications by using containers, which package applications and their dependencies into a single unit. This section covers fundamental Kubernetes concepts and the completion of a project utilizing Kubernetes.

Key Learnings:

1. **Core Concepts:**
    - **Pods:** The smallest deployable units in Kubernetes, representing a single instance of a running process in the cluster. Pods can contain one or more containers.
    - **Services:** Abstracts access to a set of pods, providing a stable endpoint (IP address and port) for accessing the pods. Services can be of different types, such as Cluster IP, Node Port, and Load Balancer.

- **Deployments:** Manages the deployment and scaling of pods, ensuring that the desired number of pod replicas are running and updating them in a controlled manner.
- **Clusters:** A set of nodes (machines) running Kubernetes, where nodes can be master nodes (controlling the cluster) or worker nodes (running applications).



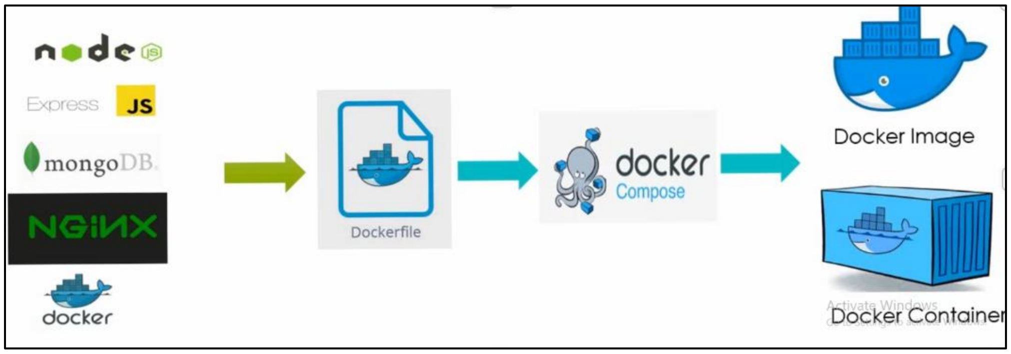
2. **Kubernetes Commands:**
   - **kubectl apply**: Applies configuration changes to the Kubernetes cluster, such as creating or updating resources.
   - **kubectl get**: Retrieves information about resources, like pods, services, and deployments.
   - **kubectl describe**: Provides detailed information about a specific resource, including events and status details.
3. **Project:**
   - **Objective:** Implement a Kubernetes-based solution to manage containerized applications.
   - **Process:** Deployed and managed containerized applications by creating Kubernetes resources such as deployments, services, and pods.
   - **Outcome:** Demonstrated the ability to automate application deployment and scaling, manage updates, and ensure high availability using Kubernetes.

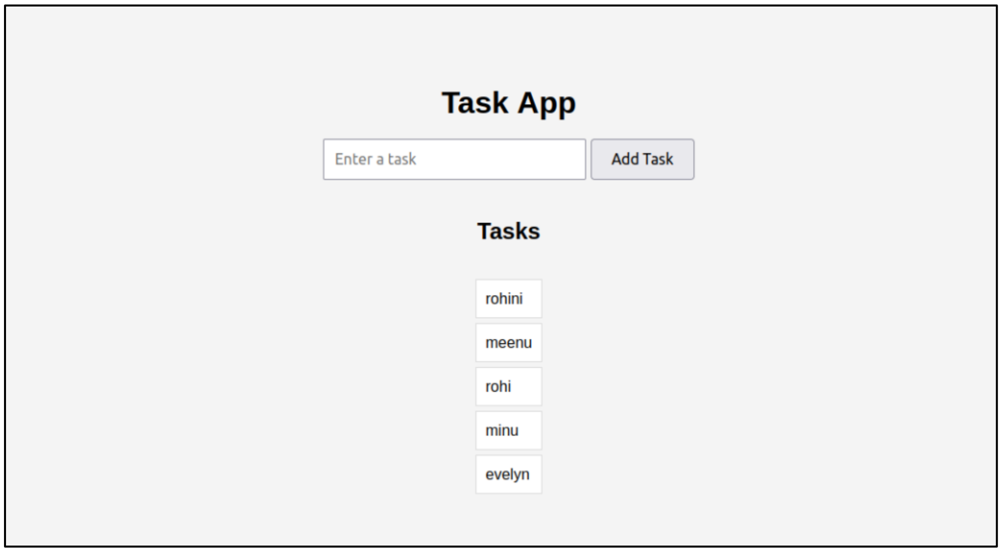
This project showcased practical skills in Kubernetes, emphasizing the platform's capabilities in orchestrating containerized applications effectively.

Project Overview:

Dockerizing a Node.js Application

1. **Project Structure:**
    - **my-task-app/**
        - **app/**: Node.js application with Express.
        - **nginx/**: NGINX configuration and Dockerfile.
        - **docker-compose.yml**: Orchestrates services.
2. **Node.js Application:**
    - **package.json**: Defines dependencies.
    - **server.js**: Main server file.
    - **Task.js**: Mongoose model for tasks.
    - **index.html**: HTML view for task input and display.
    - **styles.css**: Basic styling for the app.
3. **Docker Setup:**
    - **Node.js Dockerfile**: Builds the app container.
    - **NGINX Dockerfile**: Configures NGINX to proxy requests to the app.
    - **NGINX Config (nginx.conf)**: Routes traffic to the Node.js app.
4. **Docker Compose:**
    - **docker-compose.yml**: Defines services for Node.js app, MongoDB, and NGINX, and sets up the network and volumes.
5. **Commands:**
    - **docker-compose up --build**: Builds and starts the application, MongoDB, and NGINX services.





<u>Deployment of Dockerized Node.js with Kubernetes</u>

- `my-app/`: Root directory

- `index.js`: Main Node.js application file
- `Dockerfile`: Docker image definition
- `package.json`: Project metadata and dependencies
- `.env`: Environment variables
- `deployment-blue.yaml`: Blue deployment configuration
- `deployment-green.yaml`: Green deployment configuration
- `service.yaml`: Service configuration

Node.js Application (index.js)

Sets up an Express server to serve static files and respond with a message including a name from environment variables.

Dockerfile

Defines how to build the Docker image, including setting the working directory, copying files, installing dependencies, exposing ports, and specifying the run command.

Environment Variables (.env)

Contains configuration data such as the name to display, allowing changes without modifying code.

Package Configuration (package.json)

Metadata and dependencies for the Node.js project, ensuring consistent setup and operation.

Kubernetes Manifests
deployment-blue.yaml

Defines the blue deployment, including replicas, Docker image, and environment variables using ConfigMap.

deployment-green.yaml

Defines the green deployment for blue-green deployment strategy, allowing side-by-side deployment for testing.

service.yaml

Exposes the application via a Kubernetes service, initially pointing to the blue deployment.

Flow of Commands

1. **Build Docker Image**

```
docker build -t my-node-app .
```

2. **Start Minikube**

```
minikube start
```

3. **Set Docker Environment to Minikube**

```
eval $(minikube docker-env)
```

4. **Apply Blue Deployment**

```
kubectl apply -f deployment-blue.yaml
```

5. **Apply Service**

```
kubectl apply -f service.yaml
```

6. **Verify Deployment**

```
kubectl get deployments
kubectl get pods
kubectl get services
```

7. **Access Application**

```
minikube service my-node-app-service --url
```

Rolling Update with Blue-Green Deployment

1. **Modify and Rebuild Docker Image**

   ```
   docker build -t my-node-app .
   ```

2. **Apply Green Deployment**

   ```
   kubectl apply -f deployment-green.yaml
   ```

3. **Verify Green Deployment**

   ```
   kubectl get deployments
   kubectl get pods
   ```

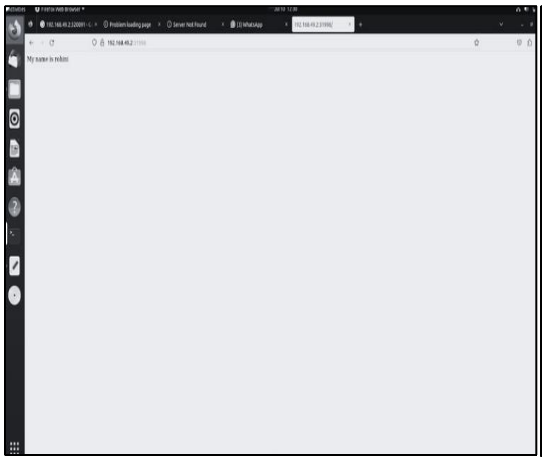
4. **Switch Service to Green Deployment**

   ```
   kubectl patch service my-node-app-service -p '{"spec": {"selector": {"app": "my-node-app-green"}}}'
   ```

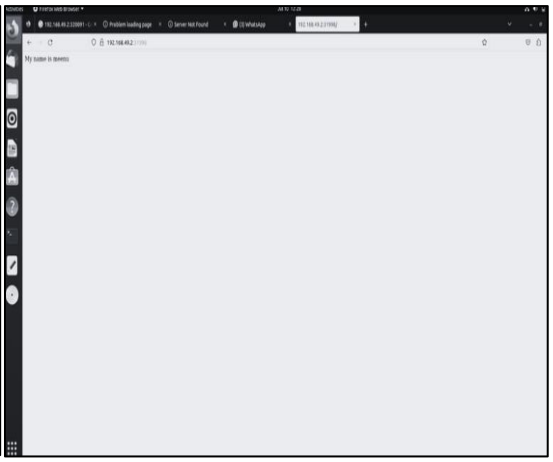
5. **Verify Service Switch**

   ```
   kubectl get services
   ```

This ensures zero downtime by switching traffic between blue and green deployments, allowing safe updates and testing.



My name is rohini



My name is meenu

CONCLUSION

The internship offered valuable insights and hands-on experience with essential cloud computing and DevOps technologies. Through in-depth training and practical projects, I gained a solid understanding of key areas such as Docker and Kubernetes, which are crucial for modern application deployment and management. Additionally, learning Linux basics, OpenStack, and monitoring tools like Grafana, Prometheus, and Loki expanded my expertise and equipped me with a well-rounded skill set. This comprehensive exposure has prepared me for a successful career in IT and cloud computing, providing a strong foundation for further professional development.

REFERENCES

- https://www.geeksforgeeks.org/cloud-computing/
- https://www.geeksforgeeks.org/linux-tutorial/
- https://www.freecodecamp.org/news/openstack-tutorial-operate-your-own-private-cloud/
- https://www.geeksforgeeks.org/docker-tutorial/
- https://www.geeksforgeeks.org/kubernetes-tutorial/
- https://kodekloud.com/
- YouTube videos for related topics

BIBLIOGRAPHY

 Docker Documentation

- Docker Inc. (n.d.). *Docker Documentation*. Retrieved from https://docs.docker.com/

 Kubernetes Documentation

- Kubernetes Authors. (n.d.). *Kubernetes Documentation*. Retrieved from https://kubernetes.io/docs/

 OpenStack Documentation

- OpenStack Foundation. (n.d.). *OpenStack Documentation*. Retrieved from https://docs.openstack.org/

 Linux Basics

- Linux Foundation. (n.d.). *Introduction to Linux*. Retrieved from https://www.linuxfoundation.org/

 VirtualBox Documentation

- Oracle Corporation. (n.d.). *VirtualBox User Manual*. Retrieved from https://www.virtualbox.org/manual/

 vi Editor

- Open Source Community. (n.d.). *The vi Editor*. Retrieved from https://www.vim.org/

 Node.js Documentation

- Node.js Foundation. (n.d.). *Node.js Documentation*. Retrieved from https://nodejs.org/en/docs/

 Nginx Documentation

- Nginx, Inc. (n.d.). *Nginx Documentation*. Retrieved from https://nginx.org/en/docs/

 MongoDB Documentation

- MongoDB, Inc. (n.d.). *MongoDB Documentation*. Retrieved from https://docs.mongodb.com/