DEPLOYMENT OF MACHINE LEARNING MODEL ON DOCKER, KUBERNETES AND JENKINS

PROJECT REPORT

ON

MLOP's PROJECT

Submitted by:

TARUSH ARYA (2110993848)

JIYA ARORA (2110993857)

TANUSH DEV (2110993870)

SAMEEP ROHILLA (2110993874)

BE CSE (ARTIFICIAL INTELLIGENCE)

GUIDED BY

Mr. Chaitanya Soni

CHITKARA UNIVERSITY INSTITUTE OF ENGINEERING & TECHNOLOGY

CHITKARA UNIVERSITY, RAJPURA

APRIL 2024

Acknowledgement

With immense pleasure **we, Mr. Tarush Arya, Ms. Jiya Arora, Mr. Tanush Dev and Mr. Sameep Rohilla** presenting **“Docker, Kubernetes and Jenkins Project Deployment”** project report as part of the curriculum of ‘BE-CSE(AI)’.

We would like to express my sincere thanks to **Mr. Chaitanya Soni** and our mentors **Dr. Kamaldeep Garg** and **Dr. Vandana Sood** for their valuable guidance and support in completing my project.

We would also like to express my gratitude towards our dean **Dr. Sushil Kumar Narang** for giving me this great opportunity to do a project on **Docker, Kubernetes and Jenkins Project Deployment**. Without their support and suggestions, this project would not have been completed.

Name: Tarush Arya (2110993848)

Signatures: ....

Name: Jiya Arora (2110993857)

Signatures: …..........................

Name: Tanush Dev (2110993870)

Signatures: …..........................

Name: Sameep Rohilla (2110993874)

Signatures: …..........................

ABSTRACT

Efficiency and reliability are key to modern software development, enabling rapid iteration, scalability, and seamless integration across multiple domains. This provides a comprehensive overview of using Docker, Kubernetes and Jenkins together to optimize project deployments.

The distribution pipeline is an essential part of any software development lifecycle, enabling the transition from development to production while maintaining consistency and reliability. Yes, Docker containers provide a lightweight, portable and easy to use environment for application packaging and deployment. By orchestrating these containers, Kubernetes automates deployment, scaling, and managements consideration; this allows for efficient use, improved scalability and fault tolerance. Jenkins is a powerful engine that complements integration and continuous delivery (CI/CD), simplifying this setup by standardizing automated tested, build, and deployment.

This report provides an in-depth look at the architecture, implementation strategies, and best practices for deploying projects using Docker, Kubernetes and Jenkins. It explores a variety of distributions, including blue-green distributions, canary distributions and rolling updates, and evaluates their suitability for different scenarios. He also discusses packaging techniques such as Dockerfile optimization and image caching to increase deployment speed and efficiency.

Additionally, the report examines the integration of monitoring and cutting equipment in pipelines to ensure visibility, monitoring performance and good problem solving. It also addresses security considerations, including containerization best practices, vulnerability analysis and access control procedures to remediate infrastructure against threats.

This report demonstrates, through real-life case studies and practical example, the benefits of using Docker, Kubernetes and Jenkins for ad-hoc deployment and compute with enhances agility, scalability and reliability. It concludes with recommendations for optimizing the pipeline, solving different problems and adapting to the changing technology landscape, enabling organizations to successfully implement the delivery process in a simple and repeatable way in today’s software development ecosystem.

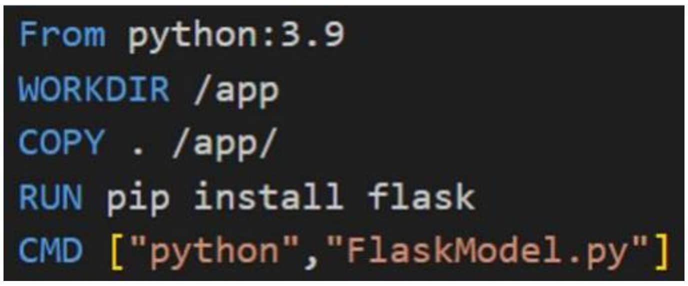
INTRODUCTION

DOCKER

Docker has transformed the landscape of software development and deployment by offering a standardized approach to package, distribute and run applications in lightweight, portable containers. Docker addresses the challenges of managing complex dependencies and ensuring consistency across different environments. Through its containerization technology, Docker encapsulates applications and their dependencies, eliminating compatibility issues and streamlining deployment workflows. Dockerfiles defines application configurations, enabling developers to create docker images that serve as portable units for running containers. This approach promotes consistency between development, testing and production environments, facilitating smoother deployment processes. Moreover, docker facilitates infrastructure as code (IaC) principles, allowing infrastructure components to be defined and manages programmatically. As organizations increasingly adopt hybrid and multi-cloud environments, Docker's portability and scalability make it a cornerstone technology in modern software development pipeline, enabling agility, efficiency and reliability.

Steps to deploy on Docker:

1. Docker Installation:
    - Visit the official Docker website
    - www.docker.com/products/docker-desktop
    - Follow installation steps
2. Make a separate folder: A.py file containing a flask model and a docker file with no extension and write the following in it.



```
From python:3.9
WORKDIR /app
COPY . /app/
RUN pip install flask
CMD ["python","FlaskModel.py"]
```

3. Image building:
    - Open docker desktop
    - Open CMD in the folder, and run the following commands.
    - Docker build -t image name.
    - The built image will be seen on docker desktop
4. Docker hub: Log in in to your docker hub online. Click on create repository and fill in the credentials
5. Image pushing to Docker Hub:
    - Run the following commands on the cmd to push the image to the docker hub.
    - Docker tag local-image:tagname new-repo:tagname
    - Docker push new-repo:tagname
6. Now our image in pushed to docker hub

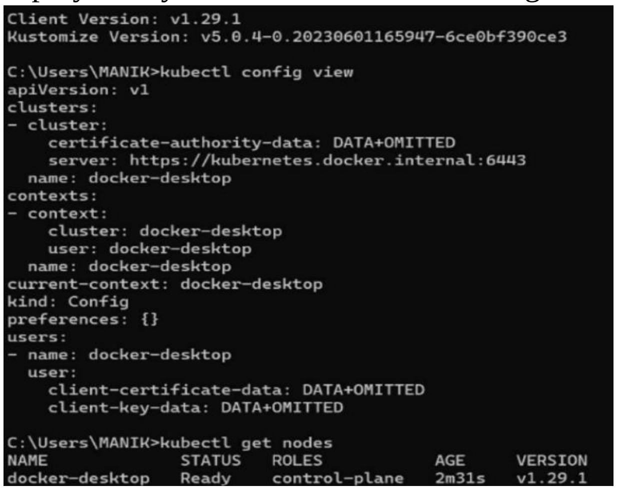
Kubernetes

Kubernetes has emerged as a pivotal technology in modernizing the deployment and management of containerized applications. Initially developed by Google and open-sourced in 2014, Kubernetes provides a powerful orchestration platform for automating the deployment, scaling and operation of application containers across clusters of machines. With the rise of microservices the architectures and containerization, Kubernetes addresses the challenges of managing complex distributed systems by abreacting away infrastructure complexities and providing a declarative model for defining desired states.

Through its robust set of features including service discovery, load balancing and automated rollouts and rollbacks, Kubernetes simplifies the management of containerized workloads, enabling organizations to achieve higher levels of agility, scalability and reliability. By leveraging Kubernetes, teams can deploy and scale applications seamlessly, optimize resource utilization, and ensure high availability in dynamic and heterogeneous environments. As organizations continue to embrace cloud-native technologies, Kubernetes has become an indispensable tool for orchestrating containerized infrastructure and driving innovation in modern software development practices.

Steps to deploy on Kubernetes:

1. Kubernetes Installation:
    - Run the following command in CMD
    - Curl.exe -LO https://dl.k8s.io/release/v1.29.2/bin/windows.amd64/kubectl.exe
    - Check the version using the following command
    - kubectl version -client
2. Docker Settings:
    - Open Docker Hub
    - Click on settings
    - Choose container terminal as integrated
    - Tick on (Expose daemon on tcp://localhost:2375 without TLS)
    - Click on apply & restart
3. Kubernetes Commands:
    - kubectl config view
    - kubectl get nodes
4. Yaml File: Build a deployment.yaml in the same folder using the following code.



```
Client Version: v1.29.1
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3

C:\Users\MANIK>kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://kubernetes.docker.internal:6443
  name: docker-desktop
contexts:
- context:
    cluster: docker-desktop
    user: docker-desktop
  name: docker-desktop
current-context: docker-desktop
kind: Config
preferences: {}
users:
- name: docker-desktop
  user:
    client-certificate-data: DATA+OMITTED
    client-key-data: DATA+OMITTED

C:\Users\MANIK>kubectl get nodes
NAME             STATUS   ROLES           AGE     VERSION
docker-desktop   Ready    control-plane   2m31s   v1.29.1
```

5. Deployments:
    - Use the following command to deploy the model
    - Kubectl apply -f deployments.yaml

Jenkins

Jenkins, a pivotal component in modern software development, serves as an automation server facilitating continuous integration and continuous delivery (CI/CD) pipelines. Originally developed as Hudson and later rebranded, Jenkins enables developers to automate tasks like building, testing and deploying applications. With its extensive plugin ecosystem and seamless integration capabilities, Jenkins streamlines workflows and enhances collaboration across development teams. By automating repetitive tasks and promoting a culture of agility. Jenkins empowers organizations to accelerate release cycles, improve code quality, and foster DevOps practices. As a result, Jenkins has become synonymous with efficient software delivery as is instrumental in driving innovation across industries.

Steps to deploy on Jenkins:

1. Log in to Jenkins using your local host.
2. Click on the new item & enter the item name to create a new pipeline.
3. Click on pipeline
4. Click on the “GitHub hook trigger for GITScm polling” and enable it.
5. Now write the pipeline script
6. Enable use Groovy Sandbox.
7. Now build the project