Project Documentation: Jenkins Pipeline for Python App Deployment

Overview

This documentation provides insights into the setup and execution of a Jenkins pipeline aimed at deploying a Python application. The pipeline incorporates various stages for testing, building, pushing Docker images to Docker Hub, updating deployment configurations, and deploying the application to a Minikube Kubernetes cluster. Additionally, the pipeline utilizes Docker Hub and Kubernetes credentials for authentication and interacts with a shared library hosted in another repository.

Pipeline Overview

The Jenkins pipeline consists of the following stages:

1. **Test Stage:**
    - Validates the codebase using automated tests to ensure code integrity and functionality.
2. **Build Image Stage:**
    - Builds a Docker image from the Dockerfile of the Python application.
3. **Push Image to Docker Hub Stage:**
    - Pushes the built Docker image to Docker Hub repository using Docker Hub credentials for authentication.
4. **Update Deployment Configuration Stage:**
    - Modifies the image name in the deployment.yaml file to reflect the newly built Docker image.
5. **Deploy to Minikube Stage:**
    - Deploys the updated application to the Minikube Kubernetes cluster using Kubernetes credentials for authentication.

Environment Configuration

- **Docker Hub Credential:**
    - Jenkins pipeline utilizes Docker Hub credentials for authentication during the image push stage.
- **Kubernetes Credential:**
    - Jenkins pipeline authenticates with the Kubernetes cluster using Kubernetes credentials to deploy the application.
- **Image Name:**
    - The Docker image name used in the deployment.yaml file is dynamically updated during the pipeline execution.

Shared Library

- **Usage:**
    - The pipeline leverages a shared library hosted in another repository for reusable functions and utilities.
    - https://github.com/ahmedmostafa56389/shared_library.git

Conclusion

This Jenkins pipeline streamlines the deployment process of a Python application by automating various stages from testing to deployment. By integrating with Docker Hub and Kubernetes, it ensures seamless authentication and deployment to target environments. The use of a shared library enhances code reuse and maintainability, contributing to the efficiency and scalability of the deployment process.