**Deployment Checklist & Guidelines**

**1. Pre-Deployment Checklist**

**1.1 Code Quality & Merge Request**

* **Code Review**: Ensure the code has been reviewed and approved by peers.
* **Branch Naming Convention**: Verify that branch names do not start with the "milestone" prefix (milestone branches are protected).
* **Merge Request Approval**: Confirm that the merge request has been approved and ready to merge with the milestone branch.

**1.2 Build Verification**

* **Clone Microservice**: Clone the microservice repository and checkout the appropriate milestone branch.
* **Gradle Build**: Run gradle assemble to generate the JAR file under build/libs.
  + Verify the JAR file is generated and available for Docker.
* **Docker file Configuration**: Ensure the Docker file is configured to use the correct JAR file and environment variables.
* **GitLab CI/CD Configuration**: Verify .gitlab-ci.yml file includes necessary stages (e.g., build, build image) and jobs for Docker image creation and push to GitLab Container Registry.

**2. CI/CD Pipeline Setup**

**2.1 Docker Image Build**

* **Ensure Docker file is Included**: Confirm the Docker file is in the repository and configured to build the image with the JAR file.
* **Verify CI Pipeline Stages**: Ensure the .gitlab-ci.yml file correctly defines pipeline stages for building and pushing the Docker image.
* **Push Code to GitLab**: Push the code to the GitLab repository.
* **Pipeline Trigger**: Ensure the GitLab CI/CD pipeline is automatically triggered on code push to the milestone branch.
* **Monitor Pipeline Execution**: Confirm the pipeline executes successfully, building the Docker image and pushing it to the GitLab Container Registry.

**3. Deployment Script Configuration**

**3.1 Modify Deployment Script**

* **Script Customization**: Update deploy\_app\_dev.sh in gitlab devops repository for the new microservice, ensuring it points to the correct Docker tags and environment variables.
* **Add Service to Script**: If deploying a new microservice, add the service name to the deployment script.
* **Environment-Specific Configurations**: Modify the script as per environment requirements (e.g., dev, QA).

**3.2 Property File Update**

* **Add Variables to dev.list**: Update environment variables in the dev.list file as required by the new microservice.

**3.3 Environment Script Setup**

* **Execute Environment Script**: Run the environment script located in the /root/.devops/ directory on the server to download necessary files, such as dev.list and deployment scripts, from GitLab to the server.

**4. Server Configuration**

**4.1 Alias Creation**

* **Create Alias for Deployment Command**: Add alias to .bashrc file:

alias deploy='/root/.devops/deploy\_app\_dev.sh'

* + This allows using the deploy command to run the deployment script.

**4.2 Volume Mapping for Logs**

* **Create Log Directory on Host**: Create /jsw/nextgen-mes/logs on the host machine to store logs.
* **Configure Log Paths in yaml files**: Update yaml file in config-library to specify the logging path /var/log/<service-name>.
* **Verify Logging Configuration**: Ensure the application writes logs to the correct location.

**5. Docker and Volume Setup**

**5.1 Dockerfile Configuration**

* **Verify Dockerfile Setup**: Ensure the Dockerfile in each microservice includes:
  + Correct environment setup for Java application.
  + Copying of the JAR file.
  + Log file configuration for persistent storage.

**5.2 Volume Mapping for Logs in Docker**

* **Attach Volume in Docker Command**: Modify the docker run command in the deployment script to map the host log directory:

-v /jsw/nextgen-mes/logs:/var/log/<service-name>