Contents

Introduction:

This document outlines a solution for a Pre-Interview Coding Challenge that involves creating a pseudo-CI/CD pipeline for a simple "Hello World" application.

The goal is to design a one-click deployment process using a tool of choice, with a preference for GitLab pipelines.

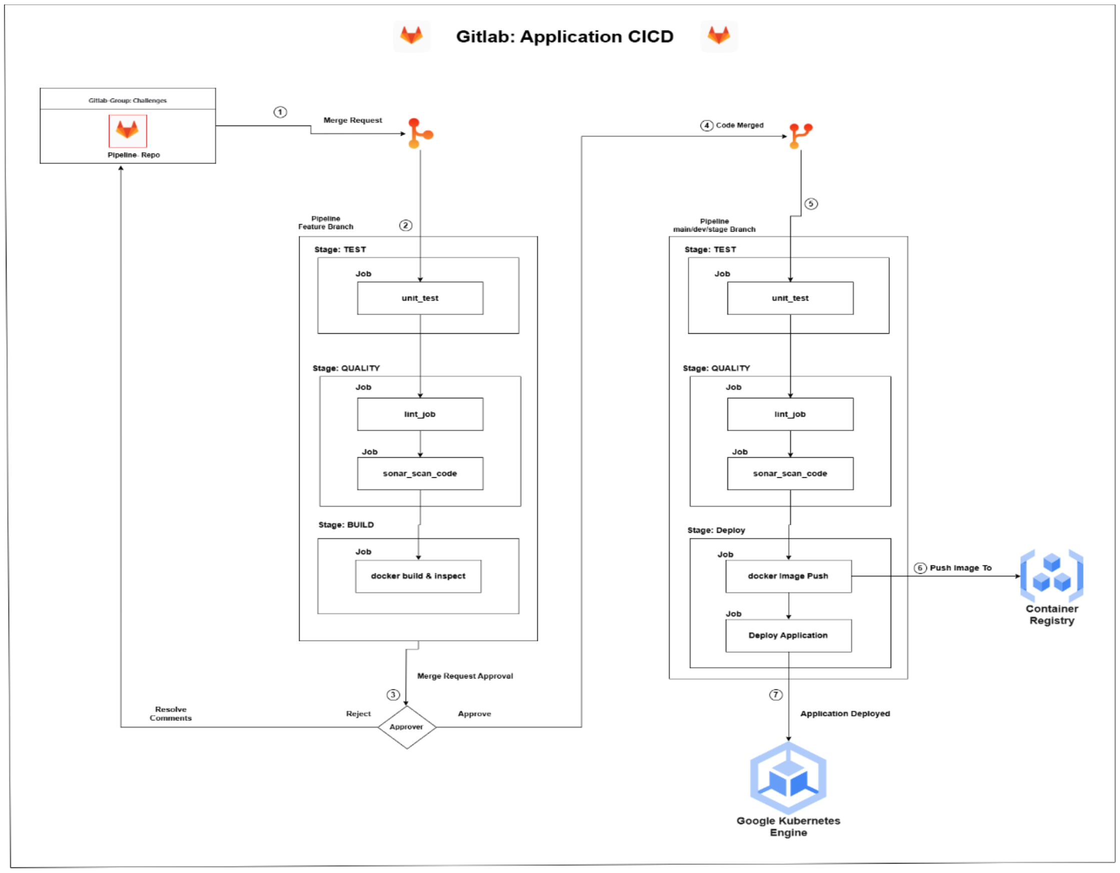
Additionally, the solution will address key considerations such as secure management of secrets, the establishment of quality gates for build validation, and the implementation of standardized deployment controls across branches.

Through this challenge, I aim to demonstrate my understanding of CI/CD principles and best practices in software deployment.

GitLab Pipeline Architecture Design:

The pipeline is designed to automate the validation, build, and deployment of the application. It starts with rigorous testing and quality checks, builds a containerized version of the application, pushes it to a container registry, and finally deploys it to a GKE cluster.

Each stage is conditionally executed based on the branch, ensuring the pipeline runs efficiently and only where needed. This setup promotes high-quality code and seamless deployments.

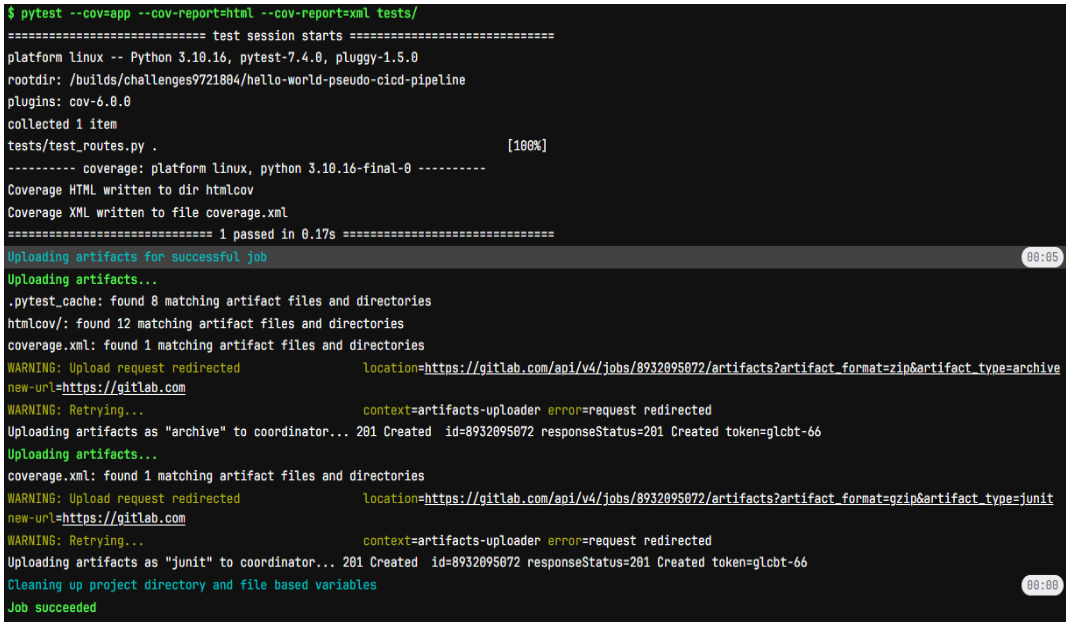


Stages and Jobs Overview:

1. **TEST Stage**
   - **Job: unit_test**

     This job focuses on running unit tests to validate the application's functionality. It uses pytest to execute the tests and generate code coverage reports in multiple formats (HTML and XML).

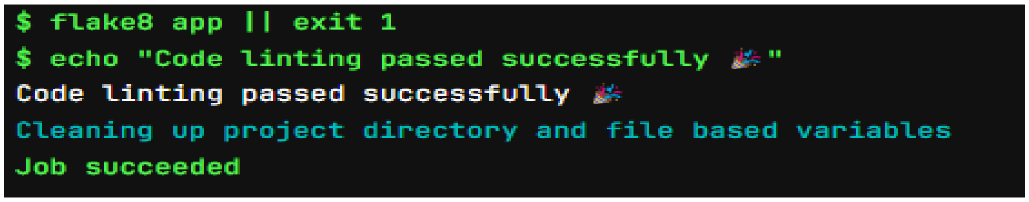
     The artifacts from this job include cached test results (.pytest_cache), HTML coverage reports (htmlcov), and an XML coverage report (coverage.xml). This job runs for specific branches or merge requests and ensures the code meets functional requirements before progressing to the next stages.



```
$ pytest --cov=app --cov-report=html --cov-report=xml tests/
============================= test session starts ==============================
platform linux -- Python 3.10.16, pytest-7.4.0, pluggy-1.5.0
rootdir: /builds/challenges9721804/hello-world-pseudo-cicd-pipeline
plugins: cov-6.0.0
collected 1 item
tests/test_routes.py .                                                   [100%]
---------- coverage: platform linux, python 3.10.16-final-0 ----------
Coverage HTML written to dir htmlcov
Coverage XML written to file coverage.xml
============================== 1 passed in 0.17s ===============================
Uploading artifacts for successful job                                                                              00:05
Uploading artifacts...
.pytest_cache: found 8 matching artifact files and directories
htmlcov/: found 12 matching artifact files and directories
coverage.xml: found 1 matching artifact files and directories
WARNING: Upload request redirected                  location=https://gitlab.com/api/v4/jobs/8932095072/artifacts?artifact_format=zip&artifact_type=archive
new-url=https://gitlab.com
WARNING: Retrying...                                context=artifacts-uploader error=request redirected
Uploading artifacts as "archive" to coordinator... 201 Created  id=8932095072 responseStatus=201 Created token=glcbt-66
Uploading artifacts...
coverage.xml: found 1 matching artifact files and directories
WARNING: Upload request redirected                  location=https://gitlab.com/api/v4/jobs/8932095072/artifacts?artifact_format=gzip&artifact_type=junit
new-url=https://gitlab.com
WARNING: Retrying...                                context=artifacts-uploader error=request redirected
Uploading artifacts as "junit" to coordinator... 201 Created  id=8932095072 responseStatus=201 Created token=glcbt-66
Cleaning up project directory and file based variables                                                              00:00
Job succeeded
```

2. **QUALITY Stage**
   - **Job: lint_code**

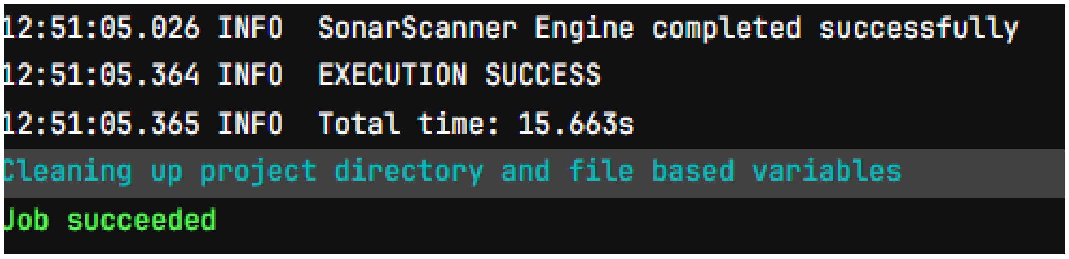
     This job ensures that the code adheres to Python coding standards by running flake8. Any linting issues are highlighted, and the pipeline stops if violations are found. This is crucial for maintaining code quality and consistency. The job is dependent on the successful completion of the unit_test job.



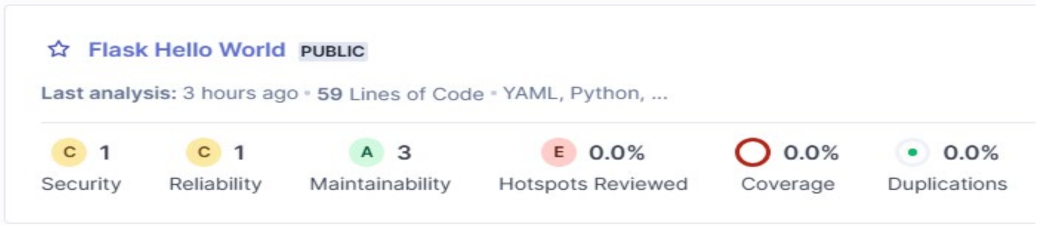
```
$ flake8 app || exit 1
$ echo "Code linting passed successfully 🎉"
Code linting passed successfully 🎉
Cleaning up project directory and file based variables
Job succeeded
```

- **Job: sonar_scan_code**

  The purpose of this job is to perform static code analysis using SonarQube. It scans the codebase for potential bugs, vulnerabilities, and code smells. This job helps identify long-term maintainability issues and ensures that the codebase remains clean and secure. It requires the successful execution of the lint_code job before starting.



```
12:51:05.026 INFO  SonarScanner Engine completed successfully
12:51:05.364 INFO  EXECUTION SUCCESS
12:51:05.365 INFO  Total time: 15.663s
Cleaning up project directory and file based variables
Job succeeded
```



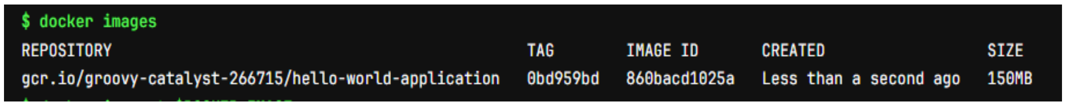
Flask Hello World PUBLIC

Last analysis: 3 hours ago • 59 Lines of Code • YAML, Python, ...

C 1	C 1	A 3	E 0.0%	0.0%	0.0%
Security	Reliability	Maintainability	Hotspots Reviewed	Coverage	Duplications

3. **BUILD Stage**
   - **Job: build_docker**

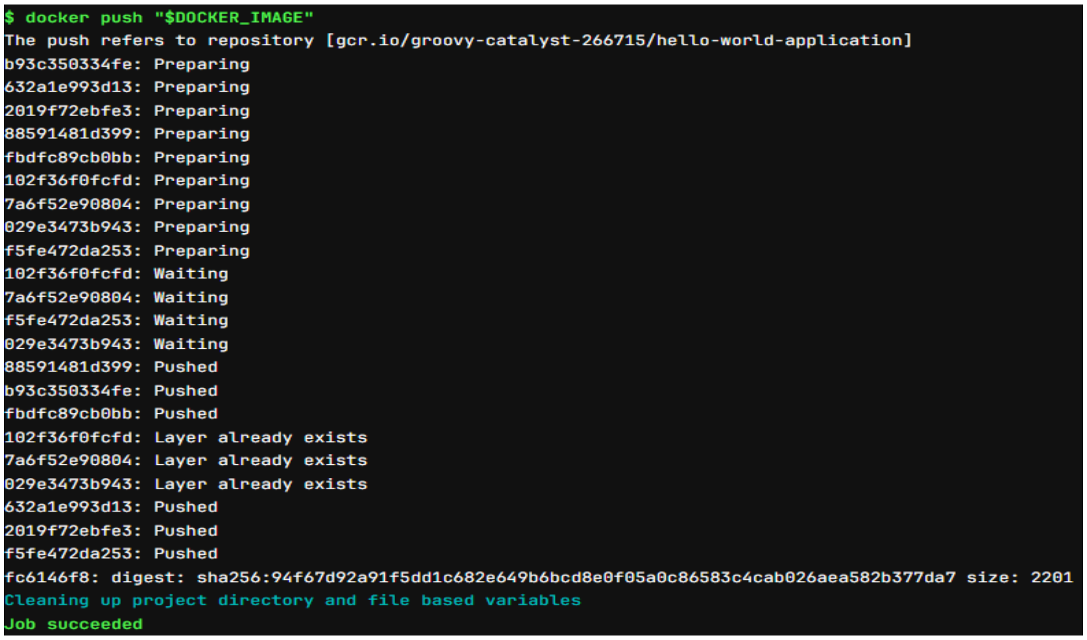
     In this job, a Docker image for the application is built using the provided Dockerfile. The job uses the docker image and the Docker-in-Docker (dind) service to perform the build. After building the image, it verifies the image by inspecting its details. This job ensures that the containerized application is ready for deployment and runs only for merge request events.



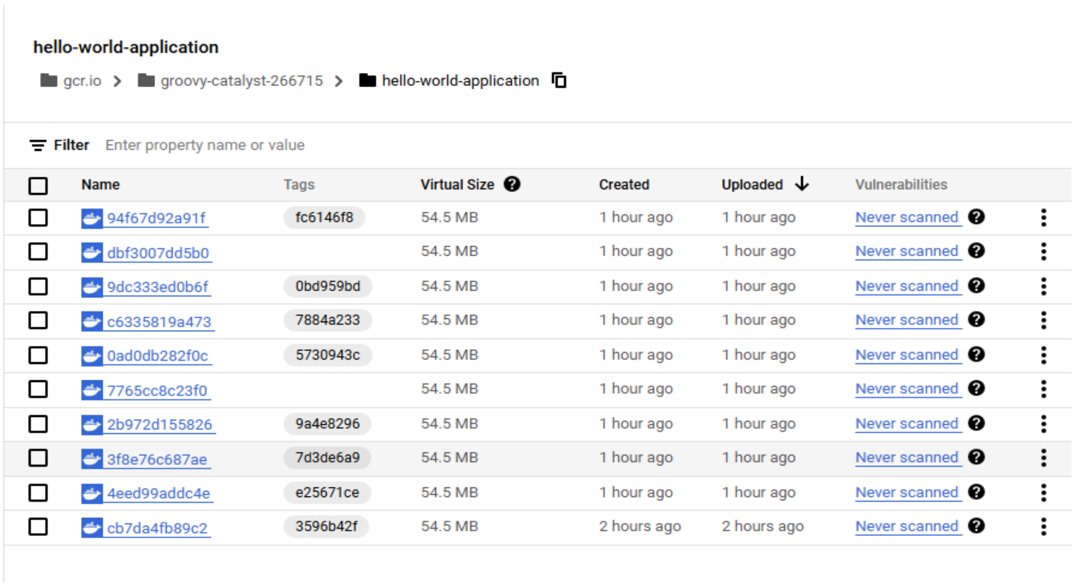
```
$ docker images
REPOSITORY                                         TAG       IMAGE ID       CREATED                  SIZE
gcr.io/groovy-catalyst-266715/hello-world-application   0bd959bd  860bacd1025a   Less than a second ago   150MB
```

4. **DEPLOY Stage**
   - **Job: push_docker**

     This job pushes the Docker image built in the build_docker stage to the Google Container Registry (GCR). It uses Google Cloud SDK to authenticate and configure Docker for GCR. Afterward, the Docker image is pushed to the registry. This step makes the container image accessible for deployment to the Kubernetes cluster.



```
$ docker push "$DOCKER_IMAGE"
The push refers to repository [gcr.io/groovy-catalyst-266715/hello-world-application]
b93c350334fe: Preparing
632a1e993d13: Preparing
2019f72ebfe3: Preparing
88591481d399: Preparing
fbdfc89cb0bb: Preparing
102f36f0fcfd: Preparing
7a6f52e90804: Preparing
029e3473b943: Preparing
f5fe472da253: Preparing
102f36f0fcfd: Waiting
7a6f52e90804: Waiting
f5fe472da253: Waiting
029e3473b943: Waiting
88591481d399: Pushed
b93c350334fe: Pushed
fbdfc89cb0bb: Pushed
102f36f0fcfd: Layer already exists
7a6f52e90804: Layer already exists
029e3473b943: Layer already exists
632a1e993d13: Pushed
2019f72ebfe3: Pushed
f5fe472da253: Pushed
fc6146f8: digest: sha256:94f67d92a91f5dd1c682e649b6bcd8e0f05a0c86583c4cab026aea582b377da7 size: 2201
Cleaning up project directory and file based variables
Job succeeded
```



hello-world-application

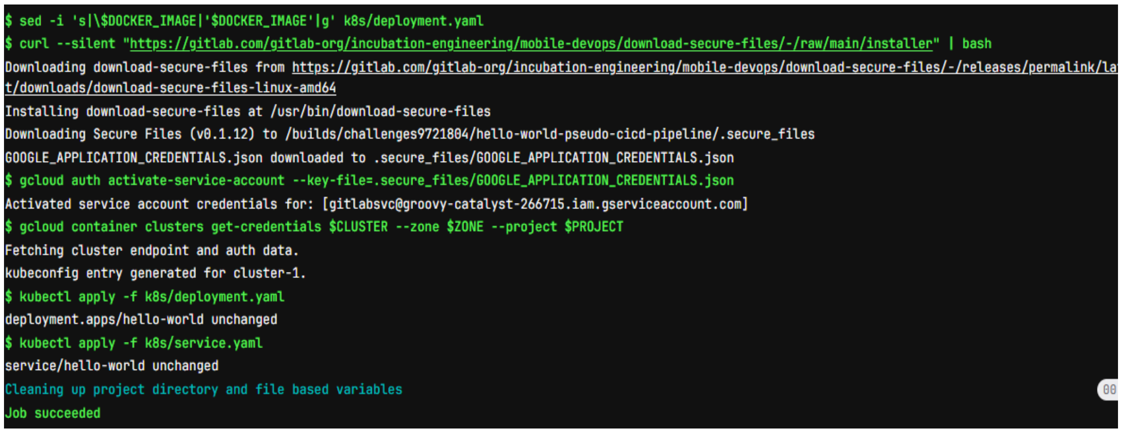
gcr.io > groovy-catalyst-266715 > hello-world-application

Filter Enter property name or value

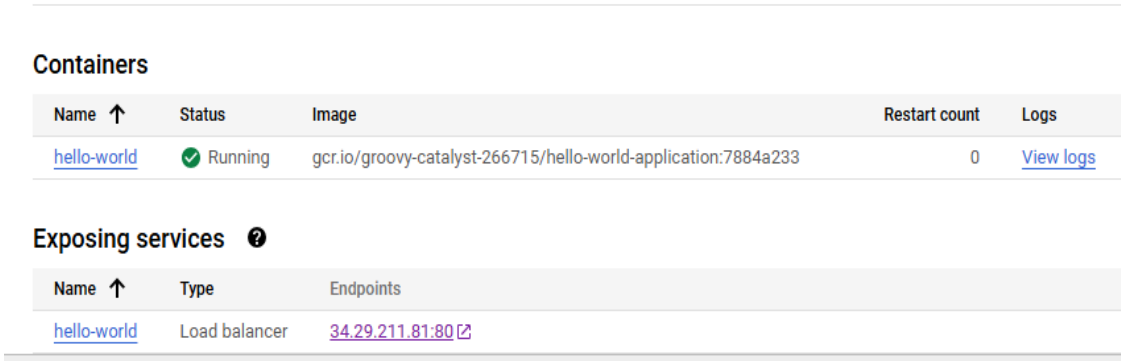
Name	Tags	Virtual Size	Created	Uploaded	Vulnerabilities
94f67d92a91f	fc6146f8	54.5 MB	1 hour ago	1 hour ago	Never scanned
dbf3007dd5b0		54.5 MB	1 hour ago	1 hour ago	Never scanned
9dc333ed0b6f	0bd959bd	54.5 MB	1 hour ago	1 hour ago	Never scanned
c6335819a473	7884a233	54.5 MB	1 hour ago	1 hour ago	Never scanned
0ad0db282f0c	5730943c	54.5 MB	1 hour ago	1 hour ago	Never scanned
7765cc8c23f0		54.5 MB	1 hour ago	1 hour ago	Never scanned
2b972d155826	9a4e8296	54.5 MB	1 hour ago	1 hour ago	Never scanned
3f8e76c687ae	7d3de6a9	54.5 MB	1 hour ago	1 hour ago	Never scanned
4eed99addc4e	e25671ce	54.5 MB	1 hour ago	1 hour ago	Never scanned
cb7da4fb89c2	3596b42f	54.5 MB	2 hours ago	2 hours ago	Never scanned

- **Job: deploy_app**

  The final job in the pipeline deploys the application to a Google Kubernetes Engine (GKE) cluster. It updates the Kubernetes deployment YAML file with the built Docker image and applies the configurations to the cluster. It uses the Google Cloud SDK to authenticate and connect to the GKE cluster. Kubernetes resources such as deployments and services are created or updated, ensuring the application is deployed and exposed to the target environment.



```
$ sed -i 's|\$DOCKER_IMAGE|'$DOCKER_IMAGE'|g' k8s/deployment.yaml
$ curl --silent "https://gitlab.com/gitlab-org/incubation-engineering/mobile-devops/download-secure-files/-/raw/main/installer" | bash
Downloading download-secure-files from https://gitlab.com/gitlab-org/incubation-engineering/mobile-devops/download-secure-files/-/releases/permalink/la
t/downloads/download-secure-files-linux-amd64
Installing download-secure-files at /usr/bin/download-secure-files
Downloading Secure Files (v0.1.12) to /builds/challenges9721804/hello-world-pseudo-cicd-pipeline/.secure_files
GOOGLE_APPLICATION_CREDENTIALS.json downloaded to .secure_files/GOOGLE_APPLICATION_CREDENTIALS.json
$ gcloud auth activate-service-account --key-file=.secure_files/GOOGLE_APPLICATION_CREDENTIALS.json
Activated service account credentials for: [gitlabsvc@groovy-catalyst-266715.iam.gserviceaccount.com]
$ gcloud container clusters get-credentials $CLUSTER --zone $ZONE --project $PROJECT
Fetching cluster endpoint and auth data.
kubeconfig entry generated for cluster-1.
$ kubectl apply -f k8s/deployment.yaml
deployment.apps/hello-world unchanged
$ kubectl apply -f k8s/service.yaml
service/hello-world unchanged
Cleaning up project directory and file based variables                                                                     00
Job succeeded
```

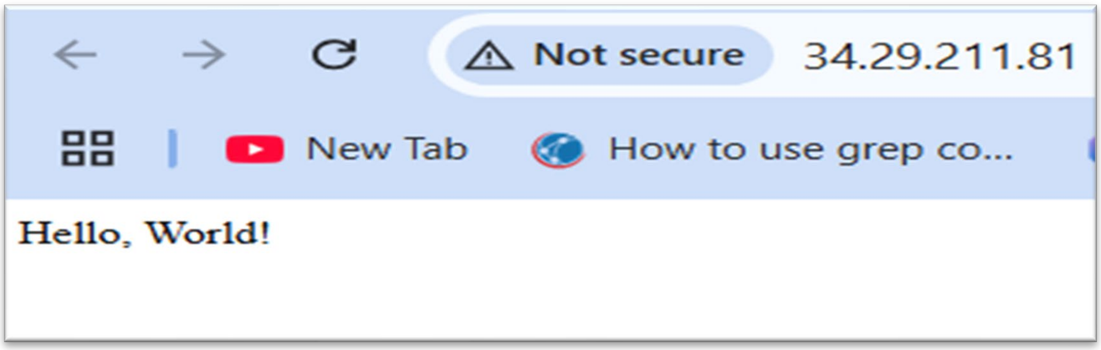


Containers

Name ↑	Status	Image	Restart count	Logs
hello-world	✓ Running	gcr.io/groovy-catalyst-266715/hello-world-application:7884a233	0	View logs

Exposing services

Name ↑	Type	Endpoints
hello-world	Load balancer	34.29.211.81:80

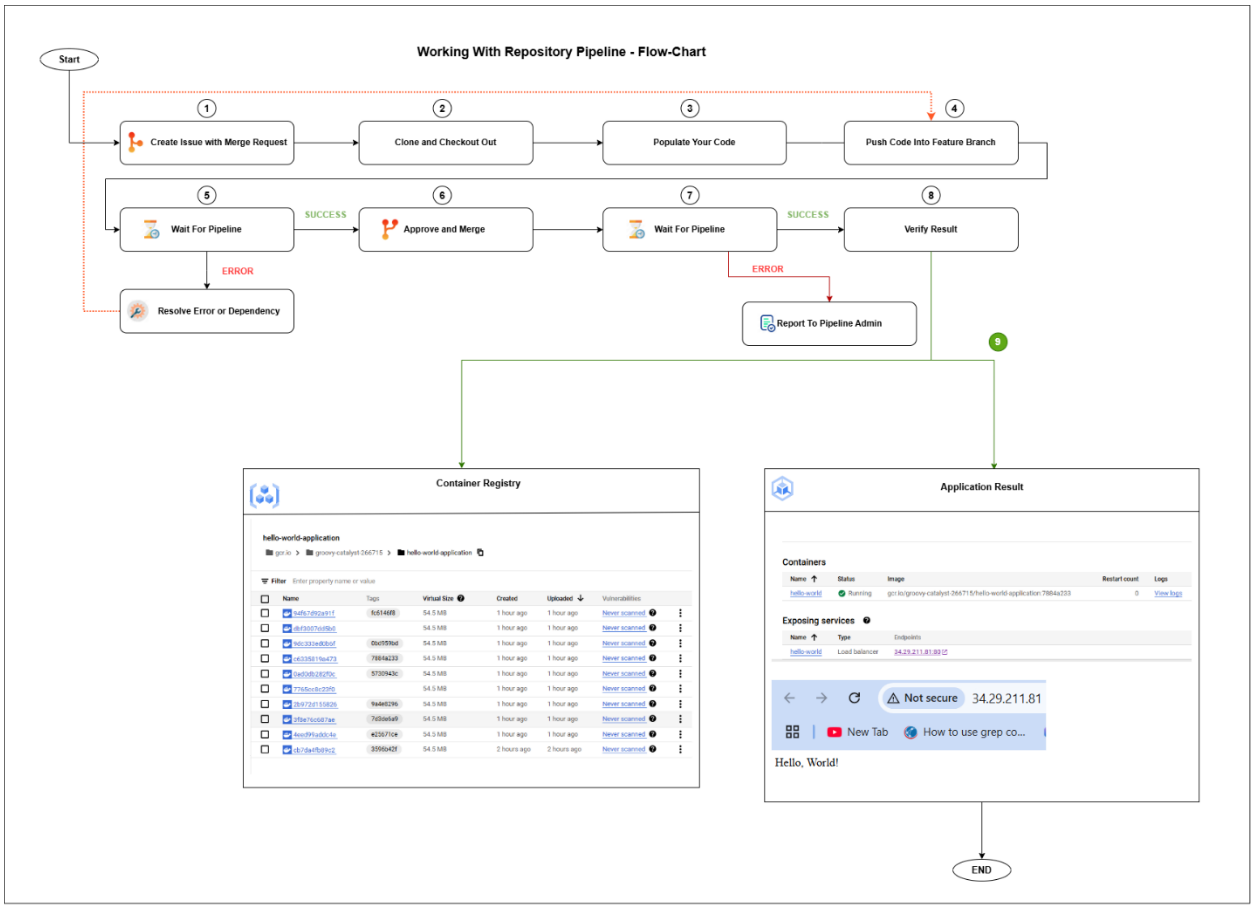


Hello, World!

Working with Repository Pipeline - Flowchart Analysis

The provided flowchart illustrates the process of working with a repository pipeline in a structured DevOps workflow.

The diagram integrates the stages of code creation, testing, deployment, and verification while accounting for error handling and reporting mechanisms. It also emphasizes the interaction with container registries and deployed application results.



hello-world-application

gcr.io > groovy-catalyst-266715 > hello-world-application

Filter Enter property name or value

Name	Tags	Virtual Size	Created	Uploaded	Vulnerabilities
94f67d92a91f	fc6146f8	54.5 MB	1 hour ago	1 hour ago	Never scanned
dbf3007dd5b0		54.5 MB	1 hour ago	1 hour ago	Never scanned
9dc333ed0b6f	0bd959bd	54.5 MB	1 hour ago	1 hour ago	Never scanned
c6335819a473	7884a233	54.5 MB	1 hour ago	1 hour ago	Never scanned
0ad0db282f0c	5730943c	54.5 MB	1 hour ago	1 hour ago	Never scanned
7765cc8c23f0		54.5 MB	1 hour ago	1 hour ago	Never scanned
2b972d155826	9a4e8296	54.5 MB	1 hour ago	1 hour ago	Never scanned
3f8e76c687ae	7d3de6a9	54.5 MB	1 hour ago	1 hour ago	Never scanned
4eed99addc4e	e25671ce	54.5 MB	1 hour ago	1 hour ago	Never scanned
cb7da4fb89c2	3596b42f	54.5 MB	2 hours ago	2 hours ago	Never scanned

Containers

Name	Status	Image	Restart count	Logs
hello-world	Running	gcr.io/groovy-catalyst-266715/hello-world-application:7884a233	0	View logs

Exposing services

Name	Type	Endpoints
hello-world	Load balancer	34.29.211.81:80

Not secure 34.29.211.81

New Tab How to use grep co...

Hello, World!

Step-by-Step Specifications

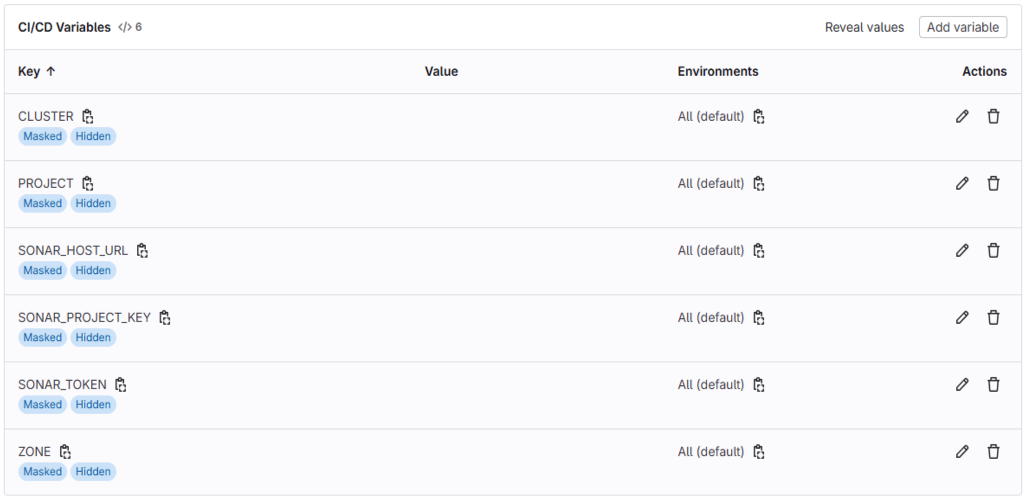
1. **Create Issue with Merge Request**
   - Begin by creating a new issue in the repository and associating it with a merge request.
   - This step ensures the task is well-defined and linked to a specific branch for development.
2. **Clone and Checkout**
   - Clone the repository to the local development environment and check out the feature branch linked to the merge request.
   - This provides an isolated space for development work.
3. **Populate Your Code**
   - Add or modify code within the checked-out branch.
   - Ensure the changes align with the requirements specified in the issue.
4. **Push Code into Feature Branch**
   - Push the updated code to the remote feature branch.
   - This triggers the pipeline to validate and test the changes.
5. **Wait for Pipeline**
   - The pipeline executes automated tests, quality checks, and builds for the pushed code.
   - Errors encountered at this stage are returned for resolution.
6. **Resolve Error or Dependency** *(if applicable)*
   - Fix any errors or dependencies identified during the pipeline execution.
   - Resubmit the code for validation through the pipeline.

7. **Approve and Merge**
   - Once the pipeline passes successfully, request approval for merging.
   - Merge the feature branch into the target branch, such as main or develop.
8. **Verify Result**
   - Validate the deployed changes by verifying the application in its environment.
   - If an error occurs during verification, report the issue to the pipeline administrator.
9. **Deployment and Result**
   - The containerized application is pushed to the container registry and deployed to the cluster.
   - Confirm the application result by checking the deployment details (e.g., container status and exposed services).

DevOps Practices Aligned with Requirements

1. Secure Way to Use Secrets

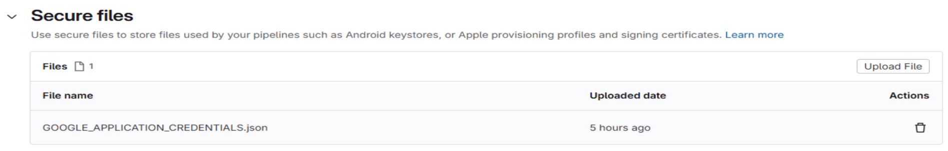
- **Using GitLab CI/CD Environment Variables**:
  - Store sensitive information, such as credentials or keys, in GitLab CI/CD environment variables.
  - Variables are **masked** to ensure their values are hidden in job logs and cannot be revealed after being saved in the settings.



CI/CD Variables 6

Key	Value	Environments	Actions
CLUSTER Masked Hidden		All (default)	
PROJECT Masked Hidden		All (default)	
SONAR_HOST_URL Masked Hidden		All (default)	
SONAR_PROJECT_KEY Masked Hidden		All (default)	
SONAR_TOKEN Masked Hidden		All (default)	
ZONE Masked Hidden		All (default)	

- **Secure Files for Sensitive Data**:
  - Use secure files to handle sensitive files, such as GCP service account key files, required in pipelines.
  - These files are securely stored and accessed during pipeline execution without being exposed.



Secure files

Use secure files to store files used by your pipelines such as Android keystores, or Apple provisioning profiles and signing certificates. Learn more

Files 1

File name	Uploaded date	Actions
GOOGLE_APPLICATION_CREDENTIALS.json	5 hours ago	

2. Quality Gate to Pass/Fail Build on Scan Results

Tools Used:

- **SonarQube**: Ensures code quality by detecting bugs, vulnerabilities, and code smells. Configured quality gates automatically fail the pipeline on threshold violations.
- **Pytest**: Runs unit tests to catch regressions, with failed tests blocking pipeline progression.
- **Flake8**: Enforces Python PEP 8 standards, treating any violations as critical to maintain code consistency.

Pipeline Workflow:

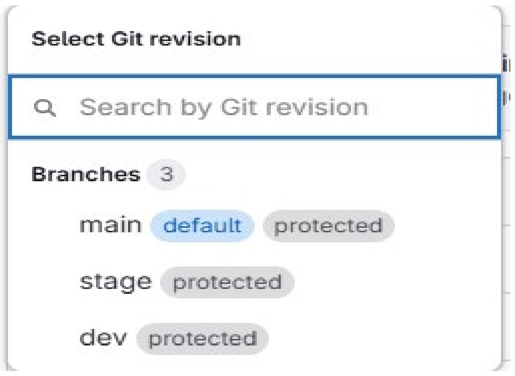
- SonarQube scans for code quality and security issues.
- Pytest validates functionality through automated unit tests.
- Flake8 ensures code adheres to style and formatting standards.

Outcome:

- Builds fail if any tool detects critical issues, ensuring secure, high-quality, and well-tested code is deployed.

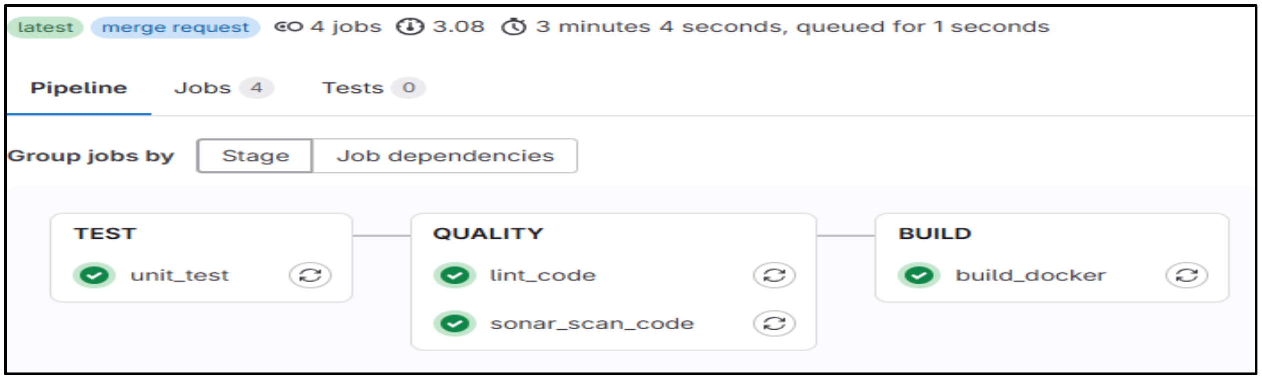
3. Standard Branch Environment Deployment Control

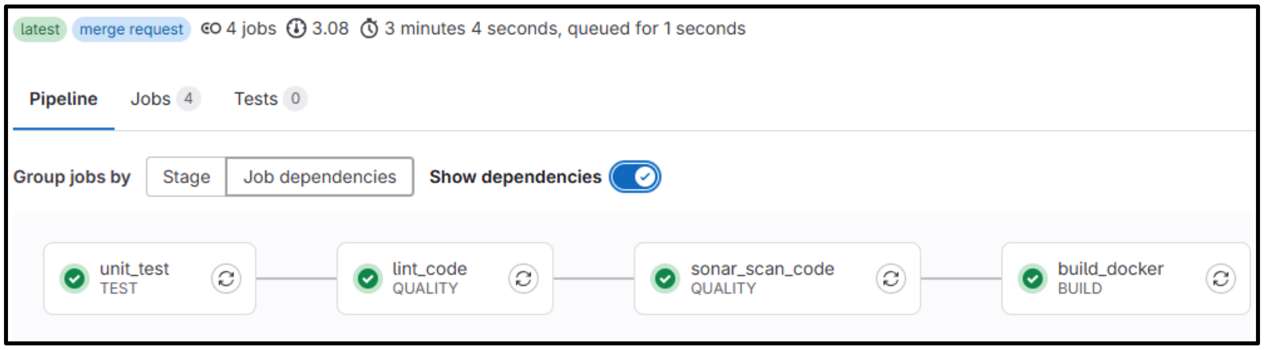
- **Branching Strategy for Environment Isolation**:
  - Maintain separate branches for each environment:
    - **dev** branch for development.
    - **stage** branch for staging/testing.
    - **main** branch for production.
  - Promote changes from one environment to another using the defined branching strategy (e.g., merge dev → stage, then stage → main).
- **Protected Branches for Controlled Deployment**:
  - Protect all environment branches (dev, stage, main) to restrict direct commits.
  - Require developers to create **Pull Requests (PRs)** for any changes to these branches.
  - Enforce code reviews and approvals before merging to ensure quality and stability.



Pipeline Standard Output for Merge Request:

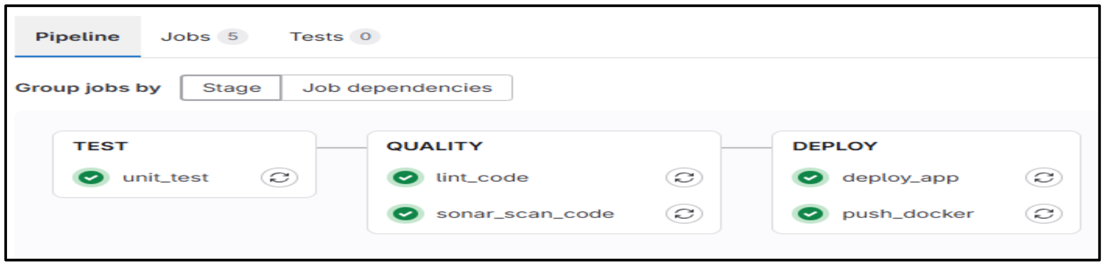
Below is the output of pipeline when running on merge request on feature branch code.

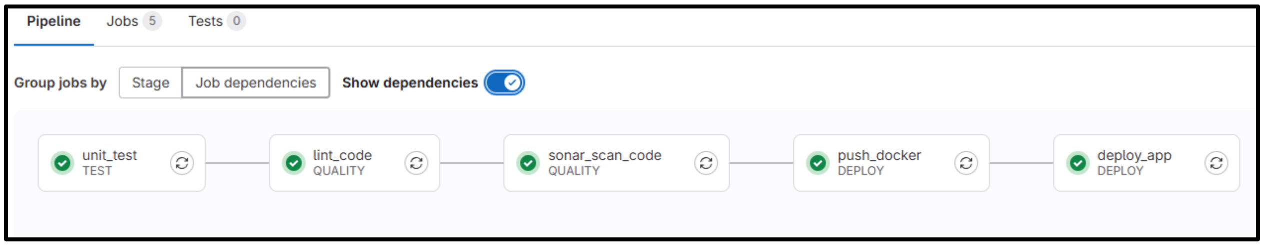




Pipeline Standard Output on Main/dev/stage branches:

Below is the output of pipeline when running on merging the code to dev/stage/main.


Pipeline
Jobs 5
Tests 0
Group jobs by
Stage
Job dependencies
TEST
unit_test
QUALITY
lint_code
sonar_scan_code
DEPLOY
deploy_app
push_docker


Pipeline
Jobs 5
Tests 0
Group jobs by
Stage
Job dependencies
Show dependencies
unit_test TEST
lint_code QUALITY
sonar_scan_code QUALITY
push_docker DEPLOY
deploy_app DEPLOY