# **CRU Testing – CI/CD Automation Process**

This document describes the **CRU (Code Review Utility) testing automation** implemented in the CI/CD pipeline. It explains the **step-by-step process**, **process flow**, and highlights the **pros and cons** of the approach.

## **Steps for CRU Testing**

1. **Refer to the Document**  
   Review the detailed implementation guide here: [Document Link].
2. **Create a Repository from Template**  
   Use the provided template repository ([Template Link]) which includes:
   * Standard folder structure
   * Pre-configured pipeline YAML
   * Required shell scripts  
     Name the repository appropriately.
3. **Edit .gitlab-ci.yml**  
   Update the following parameters:
   * **DEPLOY\_PATH**: Path to deploy the SAS code on the server.
   * **SAS\_PATH**: SAS directory path on the server.
   * **EMAIL**: Email address for pipeline notifications.
4. **Clone the Repository on Server**  
   Clone the repository into the SAS path on the server.
5. **Create a Sub-Branch**
   * Add the SAS codes in the designated **SAS directory**.
   * Place log files in the respective **log directory**.
6. **Set Script Permissions**  
   Provide executable permissions to deployment scripts:
7. chmod 744 deploy.sh
8. **Update .gitlab-ci.yml for SAS Codes**  
   List the SAS code filenames (separated by commas and spaces) in the SAS\_CODE column.
9. **Push the Test Branch to GitLab**
   * Push changes to the test branch.
   * This triggers a **Merge Request (MR)** in GitLab.
   * Assign a **maintainer** to approve the MR.
10. **Merge and Trigger Pipeline**  
    Once the MR is approved and merged into the **main branch**, the pipeline triggers automatically to test the respective SAS codes.

## **Process Flow**

The automation pipeline consists of **two stages**:

### **1. Passing All Codes and Variables**

* In this stage, all SAS codes and environment variables are passed to the pipeline.
* The pipeline zips all SAS codes and stores them in the **artifacts** for the next stage.

### **2. SA-Deploy (Deployment Stage)**

* This is the main stage where **CRU testing** is performed.
* The process includes:
  + Downloading and unzipping artifacts.
  + Passing SAS codes to CRU for validation.
  + Appending the latest logs to respective log files.
  + On successful test, SAS codes are moved to the **higher environment** using a wrapper script.
  + An email notification is sent with **pass/fail status** and respective logs.

## **Pros**

✅ **Parallel Processing**:

* Enables parallel testing and deployment of SAS codes to the next environment.

✅ **Dynamic Code Fetching**:

* Introduced an enhancement script that dynamically fetches all SAS codes, eliminating the need to manually type them into the pipeline.

## **Cons**

❌ **File Size Limitation**:

* Cannot pass multiple .sas files beyond a certain size threshold (~10–15 codes per attempt).
* Requires splitting SAS codes into smaller batches for CRU testing.
* This limitation needs enhancement at the **CRU tool level**.

❌ **Alphabetical Ordering Requirement**:

* SAS filenames in .gitlab-ci.yml must be listed in **alphabetical order**.
* If not, CRU validation fails.

# \*\*CRU Testing in CI/CD Automation - Documentation\*\*

## \*\*Overview\*\*

This document outlines the process flow, steps, pros, and cons of \*\*CRU (Code Review Utility) testing\*\* in a \*\*CI/CD automation pipeline\*\* for SAS code deployment. The goal is to automate testing and deployment while ensuring code quality before moving to higher environments.

---

## \*\*Steps to Follow for CRU Testing\*\*

1. \*\*Refer to the Documentation\*\*

- Review the provided documentation: [Document Link](#)

2. \*\*Create a Repository from Template\*\*

- Use the predefined template: [Template Link](#)

- This template includes:

- Folder structure

- Pipeline YAML (`.gitlab-ci.yml`)

- Required shell scripts

3. \*\*Configure `.gitlab-ci.yml`\*\*

- Update the following fields:

- \*\*Deploy Path\*\*: Directory where deployment occurs

- \*\*SAS Path\*\*: Server-level SAS directory

- \*\*Email\*\*: Notification email for pipeline status

4. \*\*Clone the Repository in the SAS Server Path\*\*

- Ensure the repository is cloned in the correct server directory.

5. \*\*Create a Sub-Branch and Add SAS Codes\*\*

- Create a new branch (e.g., `test-branch`).

- Add SAS codes in the `sas/` directory.

- Add logs in the respective `log/` directory.

6. \*\*Set Executable Permissions for Shell Scripts\*\*

```sh

chmod 744 deploy.sh

```

7. \*\*Update `.gitlab-ci.yml` with SAS Code Names\*\*

- Add SAS code names in the `SAS\_CODE` column (comma-separated).

- \*\*Note\*\*: Codes must be in \*\*alphabetical order\*\* to avoid CRU failures.

8. \*\*Push the Test Branch to GitLab\*\*

- A \*\*Merge Request (MR)\*\* will be created.

- Assign the MR to a \*\*maintainer\*\* for approval.

9. \*\*Merge and Trigger Pipeline\*\*

- After approval, merging into `main` triggers the pipeline.

- The pipeline executes \*\*CRU testing\*\* on the specified SAS codes.

---

## \*\*Process Flow\*\*

### \*\*Two-Stage Pipeline Execution\*\*

1. \*\*Stage 1: Passing All Codes and Variables\*\*

- All SAS codes and variables are passed.

- A \*\*ZIP artifact\*\* is created containing the SAS codes.

2. \*\*Stage 2: SA-Deploy (Deployment Stage)\*\*

- \*\*CRU Testing Phase\*\*:

- Downloads and unzips the artifact.

- Passes SAS codes to \*\*CRU for validation\*\*.

- Appends logs to respective log files.

- \*\*Post-Test Actions\*\*:

- If \*\*successful\*\*, the code moves to a higher environment via a \*\*wrapper script\*\*.

- If \*\*failed\*\*, logs indicate errors.

- \*\*Email Notification\*\*:

- Success/Failure report is sent via email.

---

## \*\*Pros of CRU Testing in CI/CD\*\*

✅ \*\*Parallel Processing\*\*

- Testing and deployment occur simultaneously, improving efficiency.

✅ \*\*Automated Code Fetching\*\*

- Dynamic script fetches SAS codes, reducing manual entry errors.

✅ \*\*Structured Logging\*\*

- Logs are systematically stored for debugging.

✅ \*\*Automated Notifications\*\*

- Email alerts keep the team informed of pipeline status.

---

## \*\*Cons & Limitations\*\*

❌ \*\*Limited Batch Processing\*\*

- CRU can only process \*\*10-15 SAS files per run\*\* (size-dependent).

- Requires manual splitting of large batches.

❌ \*\*Alphabetical Order Requirement\*\*

- SAS file names must be listed in \*\*alphabetical order\*\* in `.gitlab-ci.yml`, or CRU fails.

❌ \*\*CRU-Level Enhancement Needed\*\*

- The limitation on batch size should be addressed at the \*\*CRU tool level\*\*.

---

## \*\*Conclusion\*\*

The \*\*CRU testing automation\*\* in CI/CD streamlines SAS code validation and deployment. While it improves efficiency with parallel processing and automation, limitations like batch size and alphabetical ordering need future enhancements.

\*\*Next Steps:\*\*

- Optimize CRU to handle larger file batches.

- Remove alphabetical order dependency for smoother execution.

---

\*\*Author:\*\* [Your Name]

\*\*Last Updated:\*\* [Date]

\*\*Version:\*\* 1.0