VIEW POINT

# A 7-STEP FRAMEWORK TO IMPLEMENT CICD IN ETL TESTING

Kiran Beemanakolly,
Senior Project Manager, Infosys Limited

Vasuki Rao,
Technical Test Lead, Infosys Limited

**Abstract**

Organizations are increasingly adopting agile methodology to reduce time-to-market for product releases. However, unlike the traditional waterfall model with assigned roles, agile requires continuous integration (CI) and continuous deployment (CD) strategies for testing and deployment of code. This paper provides a 7-step framework for implementing CI and CD for ETL testing.

Infosys®
Navigate your next

## Introduction

Many organizations are looking to drive digital transformation by adopting agile methodology for Extract, Transform, Load (ETL) testing. The agile methodology mandates continuous integration (CI) to avoid manual intervention for effective test validation. It also requires continuous deployment (CD) to ensure that small increments of product releases can be delivered at regular intervals. Thus, automation becomes the key to reduce time to market.

Automating validation tasks ensures that validations are completed within sprint timelines. Further, it accelerates the process of re-validation arising from changes by minimizing manual effort spent on ensuring that the data is clean and as per the business requirements.

## How ETL validation processes work

An ETL testing workflow comprises of servers, databases and reporting tools. Servers are code repositories that store source files, intermediate files and output files and execute ETL jobs for data extraction, processing and loading. Databases are repositories containing the source and target data of ETL processing. Reporting tools such as HP-ALM, JIRA, TFS, etc., are used to update test execution status.

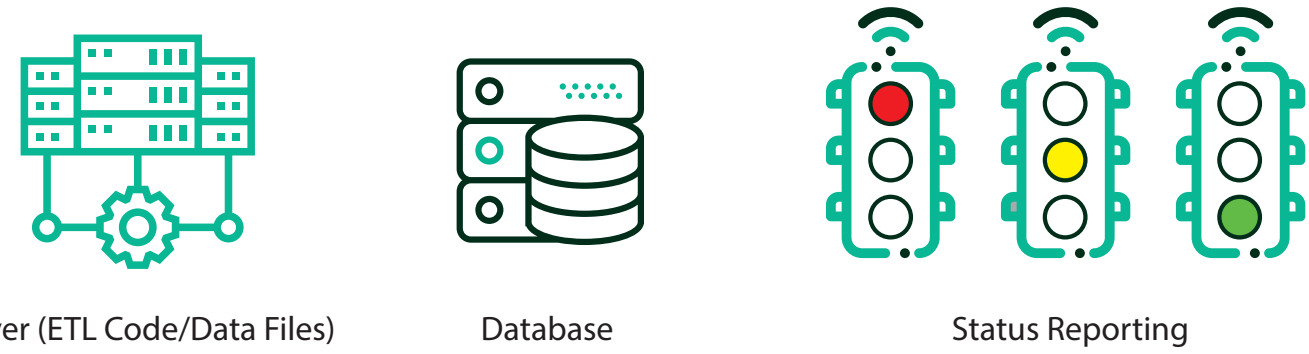



Fig 1: Components of an ETL testing workflow

Typically, ETL validation processes involve data extraction, transformation and validation that are often executed on different platforms. Thus, to automate the ETL validation process, these components must first be integrated to enable a continuous cycle of validation and deployment.

## Drivers for implementing CI/CD for ETL testing

Business drivers: Most organizations want to shift from the traditional waterfall model to agile methodology for speedy project execution and to overcome challenges such as:

- Lengthy and exhaustive testing owing to complex business rules, high data volumes and ever-changing business requirements
- Inefficient test validation arising from the need for manual intervention across the testing lifecycle
- High dependency on external teams

## Automation and CI helps businesses reduce manual intervention and accelerate time-to-market.

Technical drivers: The existing testing processes create several technical challenges such as:

- Manual execution of high number of wrapper scripts for Ab Initio ETL code takes 3 to 4 weeks
- Significant effort and time spent on rework during every sprint due to requirement changes
- Lack of automation in status reporting (HP ALM and JIRA) resulting in additional man-hours spent on updating teams about the status of each script

## Through automation and CI, testing teams can achieve 100% test coverage with zero defects within sprint timelines.

### Key requirements for iterative ETL testing

To support agile product delivery, the ETL validation steps of job execution, data validation and status reporting should be automated and integrated to run continuously as a single process, i.e., continuous integration. First, the validation steps must be interlinked to minimize manual intervention and create iterative and continuous cycles of validation. In-house or external tools can be used to create a CI workflow that automates and interlinks the validation process.

Once the code is successfully validated by the CI workflow, it must be continuously deployed in order to achieve continuous cycles of validation and deployment for agile product delivery. Here, a deployment script is created and linked with the integrator. After the CI workflow, the integrator triggers the deployment and code is automatically deployed without manual intervention. As in the case of continuous validation, in-house or external tools can be used to automate deployment, thereby ensuring iterative ETL testing.

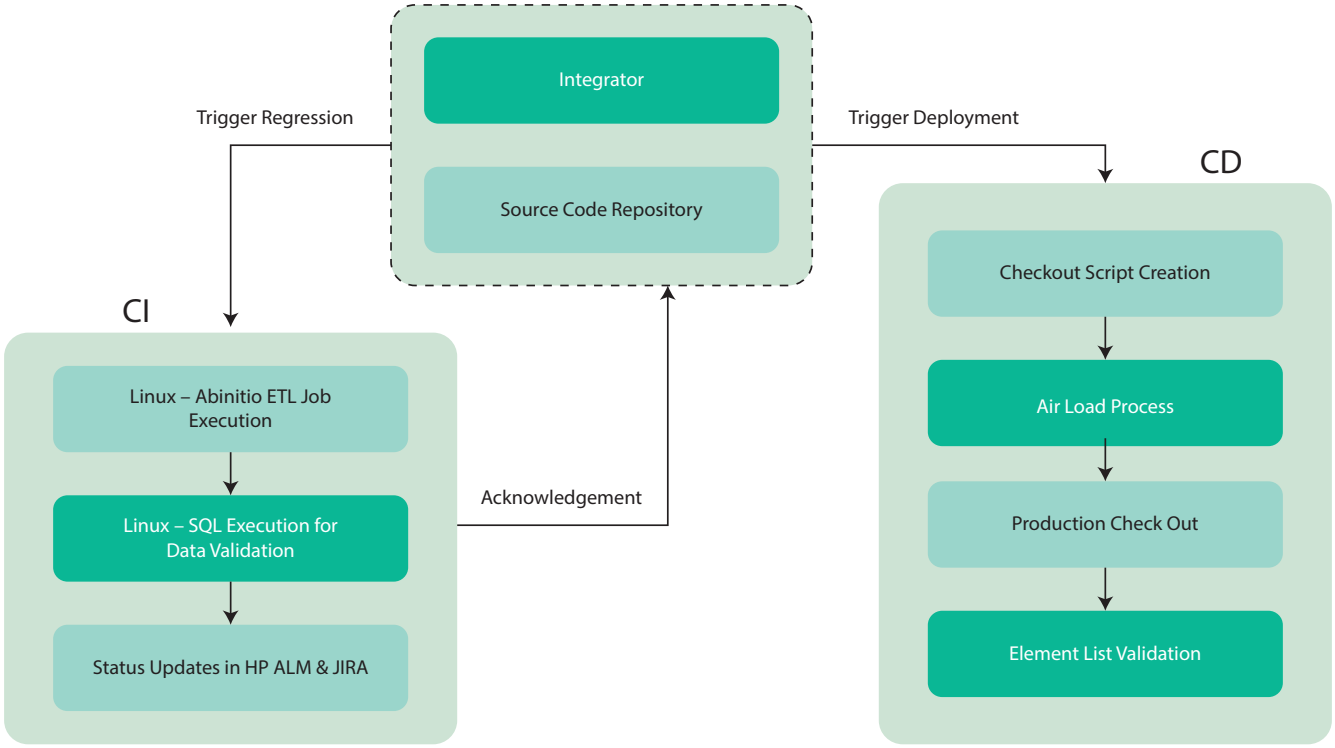



Fig 2: CI and CD workflow for continuous ETL testing

## 7 steps to implement CI and CD in ETL Testing

1. Identify an integrator
2. Determine the source code repository
3. Create an ETL code execution script
4. Define the data validation approach
5. Combine the ETL code execution and data validation
6. Enable automated status reporting
7. Create the deployment script

## Step 1: Identify an integrator

An integrator unifies all the components of ETL testing for continuous validation cycles. It acts as an interpreter by converting the output of one component into input for another component. To trigger suitable actions in each component, the output should be easily interpreted by the integrator. This requires creating job execution or data validation scripts.

The following criteria should be considered when selecting an integrator:

- Plugin availability – Different components are integrated using plugins that connect and perform the required action, making this a key criteria when choosing an integrator
- Usability – Configuring an integrator to perform the desired functionality should be simple
- Cost effectiveness – Based on the requirement, either an open source or licensed integrator can be selected

## Step 2: Determine the source code repository

The code repository stores all the ETL job execution and data validation scripts so that users can easily access the latest versions. The integrator is connected to the source code repository and, when triggered, fetches the latest version of the appropriate component (ETL job execution or validation script) from the source code repository.

In certain scenarios, the actual code can reside in the source code repository along with validation components. Here, the integrator can be programmed to execute the ETL job execution and validation wrapper whenever the code changes in the source code repository. This ensures that validations are properly triggered after code change without manual intervention.

## Step 3: Create an ETL code execution script

A wrapper script must be designed to execute ETL jobs in the right sequence. The script should be designed such that when any ETL job is aborted the status of the wrapper changes to 'fail'. This status must be communicated to the integrator so that it can generate a suitable response. In case the wrapper script status is 'pass', the integrator should trigger the next action in the workflow.

## Step 4: Define the data validation approach

It is important to build the validation script in a way that it is executed on the same server as the ETL job. This eliminates the need for manual intervention in executing validation queries on the database using any interface. Once the ETL job execution wrapper is successfully completed, the validation script must be triggered by the integrator.

In case data validation occurs in the database, the validation wrapper should connect to the database from the server, execute data validation queries and capture these results. The format of these results (whether 'pass' or 'fail') should be easily interpreted by the validation script and passed to the integrator to generate the correct response.

In cases where it is necessary to validate output/intermediate files in the server, the validation script should be executed on the server. The results from this (whether 'pass' or 'fail') should be captured in a format that is easily interpreted. These results are then passed to the integrator to generate a desired response.

## Step 5: Combine the ETL code execution and data validation jobs

Sometimes the ETL job execution follows data validation and these two steps are interlinked. Here, a combined wrapper can be created that envelops the validation wrapper as well as the ETL job execution wrapper and provides the input to the integrator. This combined wrapper is useful for regression testing where existing ETL jobs and validation queries have to be re-executed.

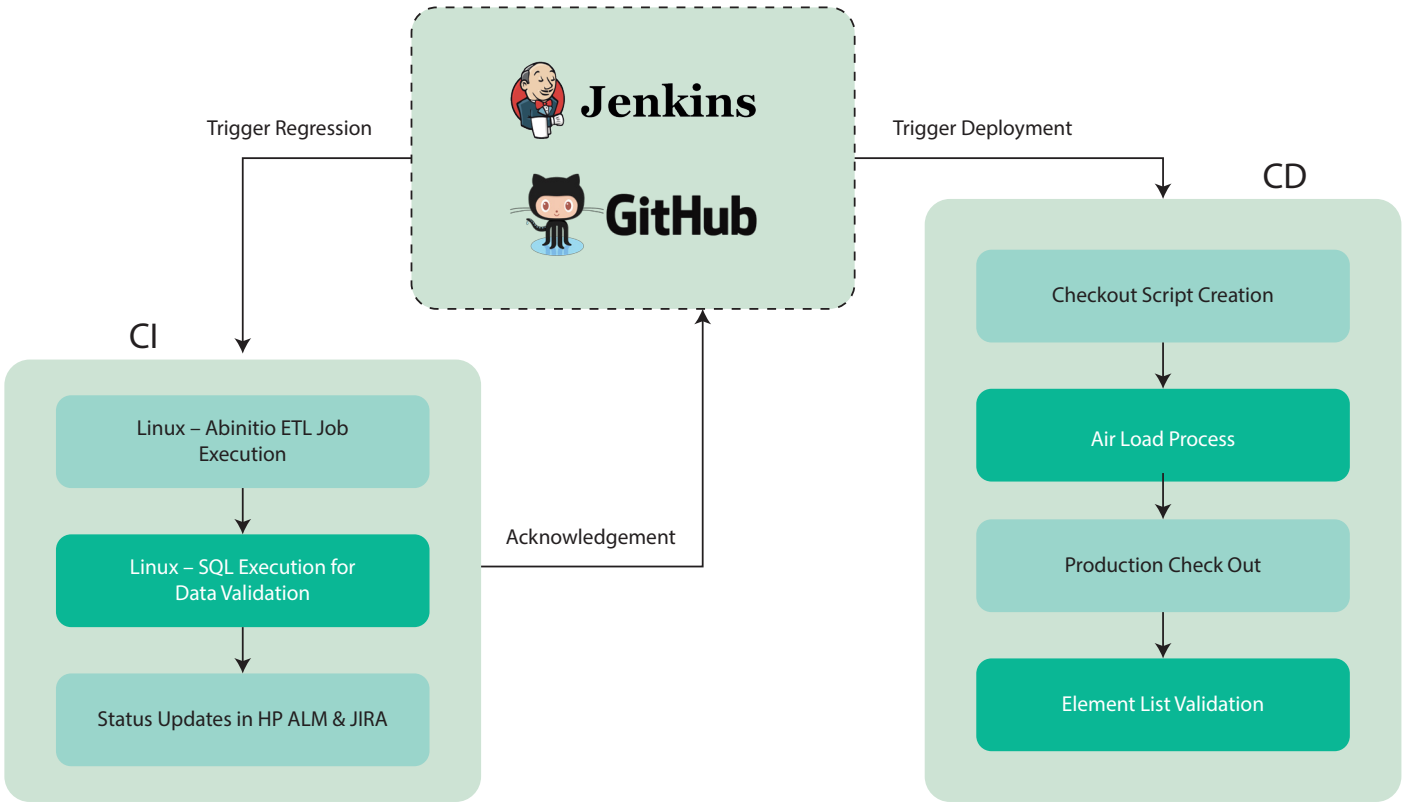
## Step 6: Enable automated status reporting

Across the CI workflow lifecycle, the execution status is updated in a test management/user stories tracking tool. This provides a real-time overview of the execution status during sprints. Once data validation is completed, the status is updated in the tracking tool. An email with the above details is sent to confirm the status for job execution and data validation.

## Step 7: Create the deployment script

A deployment script must be created and linked with the integrator such that it triggers the deployment of code without any manual intervention once the CI workflow is successfully completed.

Fig 3: Framework for implementing CI/CD in ETL testing where Ab Initio is the ETL, Jenkins is the integrator and GitHub is the source code repository.




## Benefits

Implementing CI and CD in ETL testing helps organizations:

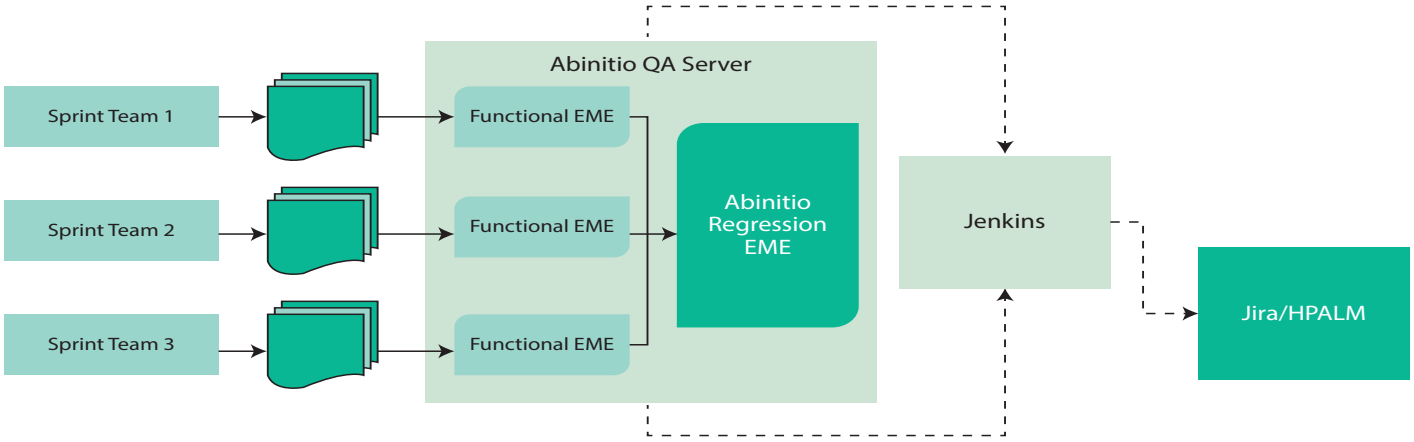
- Increase testing efficiency by automating ETL job execution, data loading, data validation, and results reporting
- Accelerate time to market by reducing turnaround time to production and as well as reduce time to deployment to minutes instead of days
- Reduce regression effort by 50%, system testing effort by 40% through iterative validation in time-boxed sprints
- Save cost by reducing the dependency on multiple external teams as execution can be triggered by any user (QCA, developer, business analyst, etc.)
- Improve quality by reducing the number of errors in code through automation

## A case study

# Automated ETL testing slashes effort by 50%

The client wanted to enable agile product delivery by shifting from the traditional waterfall model to Scrum methodology. To ensure effective test validation, they had to automate key processes, eliminate manual intervention and enable continuous integration. They needed to identify an integrator to unify the different platforms (Linux, Oracle DB, Windows). However, the traditional Jenkins plug-in was not available for Ab Initio ETL and there were connectivity issues with Jenkins and Linux QA Server. They also needed a solution to update test cases automatically using Jenkins.

Infosys used the flexible 7-step framework to implement CI and CD for agile Ab Initio ETL testing across the enterprise. The key highlights are:




- Created CI ETL framework for job execution, data load, data validation, and results reporting
- Successfully integrated Jenkins with Linux Server, HP ALM and JIRA
- Leveraged GitHub to store source scripts and connect with Linux Server
- Created a script to connect Sandbox in Linux Server via Jenkins to execute Ab-initio Graph

The flexible ETL testing framework helped the client achieve benefits such as:

- Automation of ETL job execution, data load, data validation, and results reporting
- Faster turnaround time to production with the capacity to deliver shippable increments every two weeks
- Higher validation frequency in time-boxed sprints by validating 30 user stories every 2 weeks instead of 4 weeks
- Simple execution that could be triggered by any user (QCA, developer, information analyst, etc.)
- 50% effort reduction by completely automating the regression suite through CI

## Conclusion

Even as organizations adopt agile methodology, they need to implement CI and CD across the pipeline to ensure timely and frequent product releases. The lack of CI and CD in ETL testing can lead to lengthy sprint timelines and frequent sprint failure, resulting in delayed time-to-market. The 7-step framework mentioned in this white paper provides a clear guideline on how organizations can enable CI and CD for ETL testing in agile environment without Java or Selenium. While the workflow for implementing agile may differ among teams, the overall approach remains the same. It is important to integrate all the steps in ETL validation so that a continuous and iterative workflow can be created. This ensures that validations are performed round-the-clock with minimal manual intervention, allowing organizations to deliver small and incremental releases of shippable products at regular intervals.

For more information, contact askus@infosys.com

Infosys®
Navigate your next