Test Automation Framework for Loan Delivery Assets

CS4991 Capstone Report, 2024

Nikita Jeyasingh
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
npj5kr@virginia.edu

ABSTRACT

Fannie Mae, a leading mortgage finance company, began its strategic initiative with a goal to have its Chief Data Office assets primarily covered under test automation. Recognizing the importance of accurate data and efficiency in financial reporting, I leveraged technologies such as AWS Cloud services, AWS Lambda, DBeaver, and Python to design and implement a custom-built test automation framework. This framework allowed for enhanced data reliability and streamlined testing processes. Implementing this test automation framework led to accelerated results, improved coverage, and earlier bug detection. The framework was tailored to validate incoming data efficiently, ensuring consistency and accuracy throughout the downstream vending process and aiding in the design of future frameworks for different source systems.

1. INTRODUCTION

Test automation stands as a significant cornerstone in modernizing and optimizing business processes. As financial institutions have become increasingly reliant on data-driven insights to make critical decisions, Fannie Mae set a goal for its Chief Data Office to have their assets covered under test automation. Throughout the summer of 2023, my specific focus was contributing to the Loan Sourcing Data Integration Acquisition Data Store (LSDI ADS) Team by collaborating with team members to develop a test automation framework for ADS assets.

Test automation involves the use of specialized software tools to execute predefined test cases, then compare actual outcomes with expected results, and generate detailed reports on the software's performance. Automating these tasks can accelerate the testing cycle, improve test coverage, and mitigate the risks associated with manual testing, such as human error and resource constraints. These frameworks serve as the backbone for automated testing efforts, providing a structured approach for organizing, executing, and managing test scripts and artifacts. This project sought to accelerate results, enhance coverage, expedite the detection of latent defects, and to ensure the timely delivery of high-quality software solutions.

2. RELATED WORKS

Utilization of cloud services such as AWS Lambda and AWS Step Functions played a significant role in exploring development of the custom-built test automation framework. AWS Lambda is a serverless computing service that has revolutionized the deployment and execution of code by abstracting infrastructure management. This has allowed developers to focus solely on the execution of code (Amazon Web Services, n.d.) This offers good scalability, increases cost-efficiency, and simplicity making this a desirable choice for integrating test automation into cloud-based workflows.

Additionally, AWS Step Functions, a serverless orchestration service, complements AWS Lambda with the creation of complex workflows and state machines. This capability allows for the orchestration of end-to-end test automation workflows which can encompass multiple stages, environments, and dependencies (AWS Step Functions, n.d.). AWS Step Functions allow for sequential or parallel execution flows to be defined, handle error conditions, and integrate jobs with various other AWS services, which allows for the integration of test automation into broader software delivery pipelines. The serverless nature of both the AWS Lambda and Step Functions offers advantages in terms of cost optimization, as teams only pay for the resources consumed during test execution. The integration of AWS Lambda and Step Functions allowed for this framework to be built and offered a pivotal role in the design and implementation process.

3. PROJECT DESIGN

A systematic approach was necessary to implement and integrate the custom-built test automation framework within Fannie Mae's technological ecosystem. Emphasizing seamless integration, this project required meticulous planning, collaboration with various teams, and adaptation to existing systems to ensure a smooth deployment process.

3.1 Review of System Architecture

The custom-built test automation framework is architecturally designed to integrate seamlessly with AWS Cloud Services to facilitate data processing and testing automation. The high-level architecture encompasses:

- Driver Lambda: Serves as the point for the framework, processing input JSON to trigger specific test types and data sets.
- Test Orchestration: Utilized AWS Step Functions to manage the flow of testing processes, including application code execution, test data submission, and test result analysis.
- AWS Services Integration: Services such as: Lambda, S3, RDS, ECS, and SNS, are orchestrated to handle various aspects of the testing process, from data storage to job execution and notifications.

3.2 Requirements

The Chief Data Office at Fannie Mae set a goal for their assets to be primarily covered under test automation. Fannie Mae's Loan Sourcing Data Mart (LSDM) is a reporting database consolidating multiple Single Family Data sources. LSDM was previously deployed on-premises as the trusted source for business reporting and operations and was previously deployed on-premises. The Loan Sourcing Data Integration (LSDI) Team was part of a collective effort to re-architect and migrate LDSM to AWS services for each of the following source systems: Delivery, Cash Acquisition, MBS Acquisition, Purchase ECF, Funding ECF, NCD, and Uniform Closing Disclosure Data (UCD). Each source system contains data and messages that are published to queues that come from upstream vendors.

The framework for this project was tailored specifically to the Cash Acquisition source system, which refers to the data produced when Fannie Mae acquires a loan after a loan is certified from a lender that funds that loan. The end-goal is to have the test automation run in a controlled environment that mimics the source system.

3.3 Key Components

The custom-built framework leverages the use of AWS services, each step essential to the orchestration, execution, and management of tests. AWS Lambda functions act as the drivers of this process, triggered by various mechanisms including manual initiation, CloudWatch events, and Jenkins pipelines. Upon receiving input in the form of JSON, the functions begin a series of orchestrated test procedures managed by AWS Step Functions.

The framework's key components are defined by their role in the automation process:

- AWS Lambda: Serves as the core execution environment for running code in response to events, such as new data submissions or test trigger requests.
- AWS Step Functions: Manages the orchestration of testing workflows, linking multiple AWS services and Lambda functions into a cohesive process.
- Amazon Simple Notification Service (SNS): Utilized for publishing messages to trigger actions across the system, such as notifying relevant stakeholders of test starts, completions, or failures.
- Amazon Simple Storage Service (S3): Serves as the central repository for storing test cases, data sets, and test results.
- Amazon Relational Database Service (RDS): Stores and manages structured test result data, allowing for analysis and reporting of test outcomes.
- Amazon Elastic Container Service (ECS): Manages containerized applications, facilitating the execution of test environments that require specific configurations or dependencies.

3.4 Framework Implementation

The system initiates testing workflows through three primary mechanisms: manual triggers, automated scheduling via AWS CloudWatch, and integration with CI/CD pipelines through Jenkins.

3.4.1 Driver Lambda Function

Upon initiation, the Driver Lambda function is invoked, serving as the entry point to the framework. The Driver Lambda function parses the input JSON, the function that reads and parses input JSON data, which contains vital parameters for the test execution such as: source name, test type (regression, smoke, progression), component to be tested, test case version, and test data version. Based on the parsed information, the Driver Lambda sets up the initial conditions for the test, determining which components will be tested and used.

3.4.2 Test Orchestration with AWS Step Functions

Following the setup initiated by the Driver Lambda, the Test Orchestration phase is broken down into steps, managed as states within the AWS Step Functions state machine:

- Application Code Execution: Executes the specific application code or testing scripts relevant to the chosen test type and component.
- Check Process: Verifies that the application code execution step was successful and that the initial conditions for further testing are met and ensures that the test can proceed as planned without carrying forward any errors.
- Publish Messages: Utilizes Amazon SNS to publish messages regarding the status of the testing process used for notifying stakeholders for taking further actions within the framework, or logging purposes.
- ETL (Extract, Transform, Load) Jobs: Prepares the test data by extracting data from specified sources, transforming it according to the test requirements, and loading it into the target system or database.

3.4.3 DATA & TEST MANAGEMENT

Following the execution of the ETL jobs, the process transitions ensure the validation of test results. In the following stages, the framework manages the submission of test data and collects results:

- DTF (Data Test Files) Submission: Submitted and managed throughout the test execution process, including preparing data in Amazon S3, and managing metadata in Amazon RDS.
- Test Results: As tests are executed, successes, failures, and detailed logs are

stored in Amazon S3 for raw results and Amazon RDS for structured result data, enabling a thorough analysis of the outcomes.

4. RESULTS

The test automation framework for the Cash Acquisition source system was successfully designed and is currently in its final stages of production deployment. Cash Acquisition, one of the seven source systems identified for automation, has served as a foundational model for the development of subsequent five remaining test automation frameworks.

In the final stages of testing and validation, the Cash Acquisition test automation framework has demonstrated its capability to perform end-to-end testing successfully, including validation of data integrity, performance assessments, and the automated detection of anomalies.

The final phase of the process involves analyzing the collected test results and generating reports. This step allows for developers to understand the quality and reliability of the software being tested. The framework may integrate with additional tools or services for visualizing test outcomes, identifying trends, and making data-driven decisions regarding the software's readiness for production deployment.

5. CONCLUSION

The development of the Test Automation Framework at Fannie Mae represents significant advancement in the integration of automated testing for the company. This project has demonstrated the utility of AWS Cloud services, particularly AWS Lambda and AWS Step Functions, to create a scalable testing environment that enhances data reliability and efficiency in financial reporting. The framework has successfully reduced the time and resources required for testing, improved coverage, and enabled earlier detection of discrepancies, ensuring data integrity and compliance with stringent financial regulations.

This framework streamlines the testing process and lays solid foundation for future enhancements and integrations for alternative automation frameworks. Its adaptability to different testing scenarios and potential to reduce manual intervention aids the company's ongoing efforts to ensure technological excellence.

6. FUTURE WORK

The current framework addresses the needs of the Cash Acquisition source system; there is substantial scope for its application to other source systems within Fannie Mae. Future work will focus on adapting the framework to additional source systems, enhancing its scalability and versatility. Potential improvements include integrating advanced data analytics features and exploring the use of machine learning to predict and rectify anomalies before they affect the system. Continued development will ensure the framework remains at the forefront of technological advancements and can meet the evolving demands of the financial industry.

REFERENCES

Amazon Web Services. (n.d.). What is AWS Lambda? Developer Guide. Retrieved from https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

AWS Step Functions. Serverless. (n.d.). https://www.serverless.com/guides/aws-step-functions