MAINFRAME BANKING APPLICATION CASE STUDY

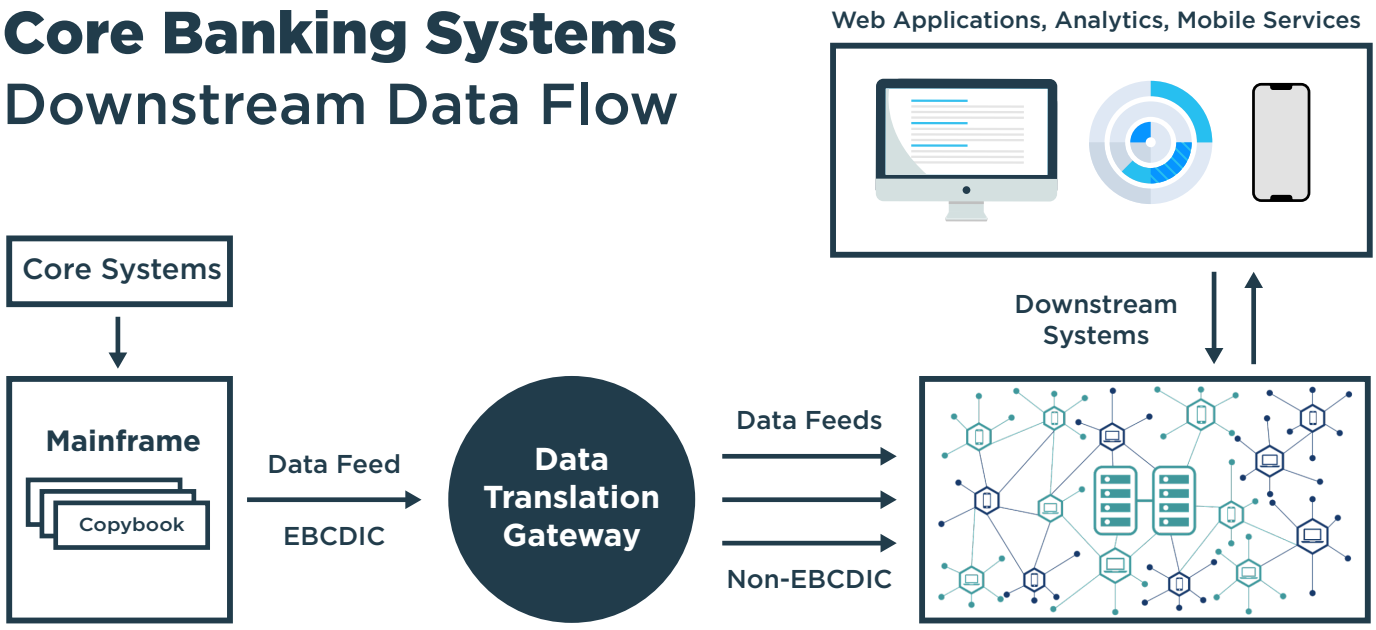
Synthetic Data Enables Tech Transformation at Global Bank

A large and rapidly growing multi-national bank operating in 32 markets is currently providing banking and financial products and services to over 8.5 million retail and business customers. The IT organization at the bank employs thousands of engineers who maintain hundreds of application systems and are geographically distributed across its worldwide operations.

Like many financial institutions, the bank is investing heavily in digital transformation and improving its use of data and analytics to deliver a superior digital banking experience to its customers. The bank operates its core banking applications on mainframe systems that feed approximately 500 downstream systems representing a variety of applications for all lines of business within the bank.

The Test Data Management (TDM) team discovered GenRocket while trying to solve a tough software testing challenge – They wanted to **generate synthetic data in IBM EBCDIC format based on a COBOL Copybook file structure and provision synthetic test data at scale.** Their objective was to conduct end-to-end testing of batch data flows from their long-standing mainframe environment to newer distributed systems developed as part of the technology transformation initiative. End-to-end system testing is essential for validating the integrity of data as it flows between diverse systems across all areas of the bank's IT infrastructure.

Previously, the bank had to rely on EBCDIC test data provisioned from the production environment. This required a cumbersome process to requisition, copy, mask, and subset the data for testing and was not considered a scalable approach. That's because the bank maintains over 5,000 COBOL copybooks that represent wide-ranging file structures and data domains. **Copybooks define the data that flows between mainframe systems and a centralized data translation gateway that reformats the data for consumption by downstream systems.**



The test data team needed EBCDIC test data with high variety and in high volume to perform end-to-end testing across this complex environment. To validate the integrity of **all** connected systems and their many data flows, EBCDIC test data is needed for **all** file structures and formats. **GenRocket has proven to be an ideal solution to meet this challenge. It's Test Data Automaton (TDA) platform can generate realistic synthetic data in any data format, with complete control over data variety and volume while maintaining full referential integrity.**

The EBCDIC Test Data Generation Challenge

Generating synthetic EBCDIC data is a tough challenge because its structure and encoding scheme is completely different from modern file formats like CSV, XML, or JSON encoded with ASCII or Unicode character sets. EBCDIC is derived from the original data format used by IBM mainframe peripherals (e.g., disk drives, tape devices and card readers) and uses 8-bit character encoding with fixed-length fields to define its data elements. This is in sharp contrast with the variable length records and delimiting characters found in today's most used data formats.

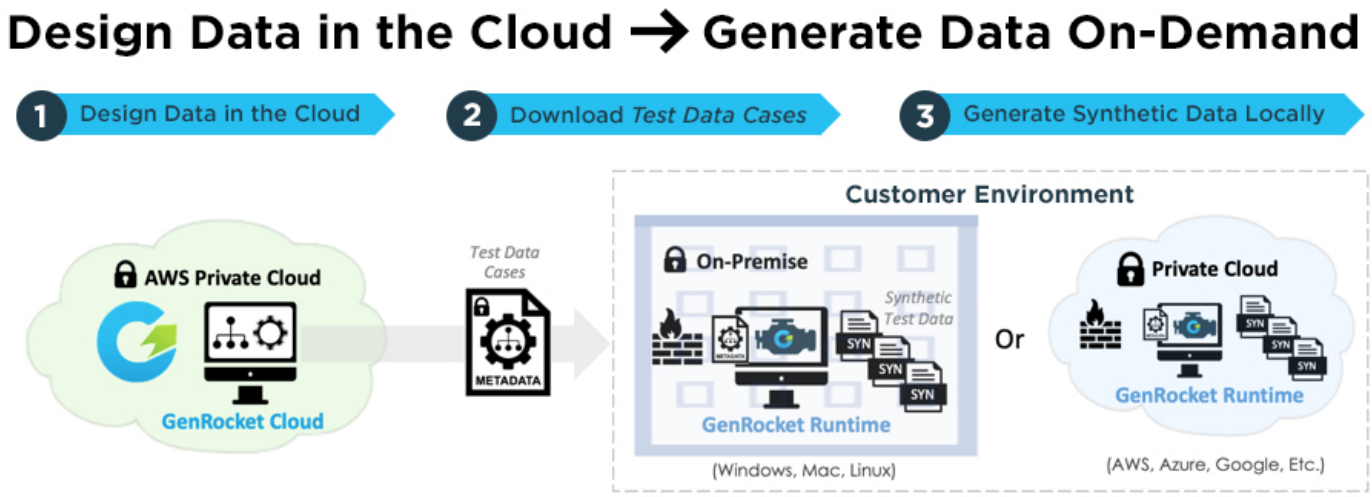
Additionally, COBOL copybooks often include clauses that REDEFINE some fields based on the value of other fields or use the OCCURS/DEPENDING_ON clause to define variable length arrays. This presents a difficult data translation challenge, but one that is performed with speed and efficiency by the bank's data translation gateway.

The challenge for synthetic data generation is to accurately simulate EBCDIC data flows based on the data models defined by thousands of copybooks while ensuring the accuracy and validity of the synthetic data being generated.

Prior to GenRocket, there was no synthetic data platform capable of generating test data using the EBCDIC file format, especially with the speed and control over data variety offered by its Test Data Automation platform. GenRocket's ability to import COBOL copybooks containing the EBCDIC data model directly into its platform, automatically assign generators from its intelligent data warehouse, and apply rules for controlling data volume and variety provides an easy and automated approach to synthetic EBCDIC data generation.

Flexible Cloud Deployment and DevOps Integration

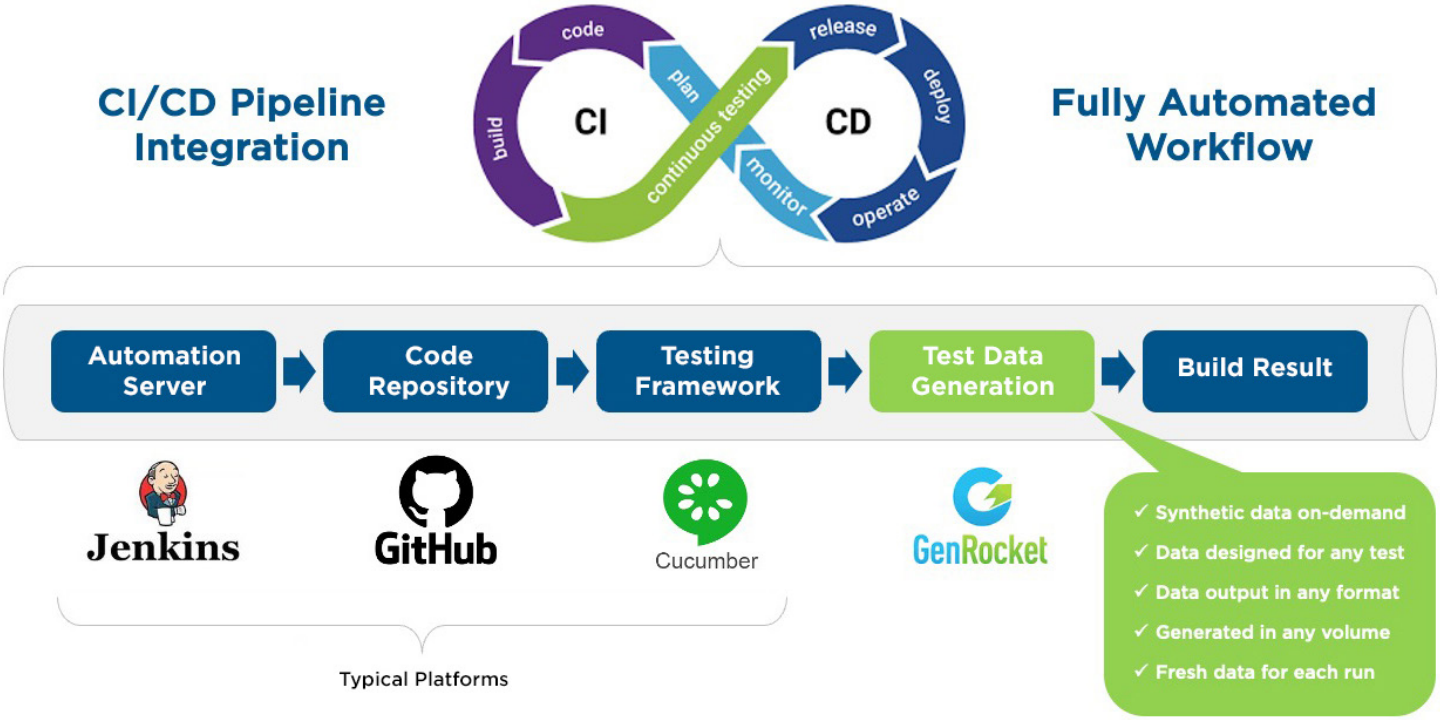
Hybrid cloud solutions like GenRocket's *Test Data Automation* (TDA) self-service platform offer the ease and agility needed by the bank for the era of cloud-base testing. With GenRocket, DevOps teams collaboratively design the exact data needed for testing in the ***GenRocket Cloud,*** a secure SaaS solution hosted on an AWS private cloud. Once designed, a template for the required test data is downloaded, in the form of a ***Test Data Case.***



This template is a secure encrypted file containing only the metadata required to generate test data using the ***GenRocket Runtime Engine.*** This engine can be integrated with any test automation tool and triggered to generate synthetic data in real-time during test execution using a physical machine on the customer premise, or a virtual machine in the customer's private cloud.

The GenRocket *Runtime Engine* supports the virtualization environments of all major cloud platforms including Amazon Web services, Microsoft Azure, and Google Cloud Platform. Additionally, the *GenRocket Runtime* Engine and its resources can be packaged as a single portable image and deployed in a container such as Docker.

It's easy to integrate GenRocket with any test automation tool and deploy its TDA technology into a fully automated release pipeline as illustrated below.



There are many ways that GenRocket can integrate into CI/CD pipelines within an Azure, Google Cloud, Amazon Web Services, or other cloud-based environment. GenRocket *Test Data Cases* and *Scenarios* can be launched using any of the following methods:

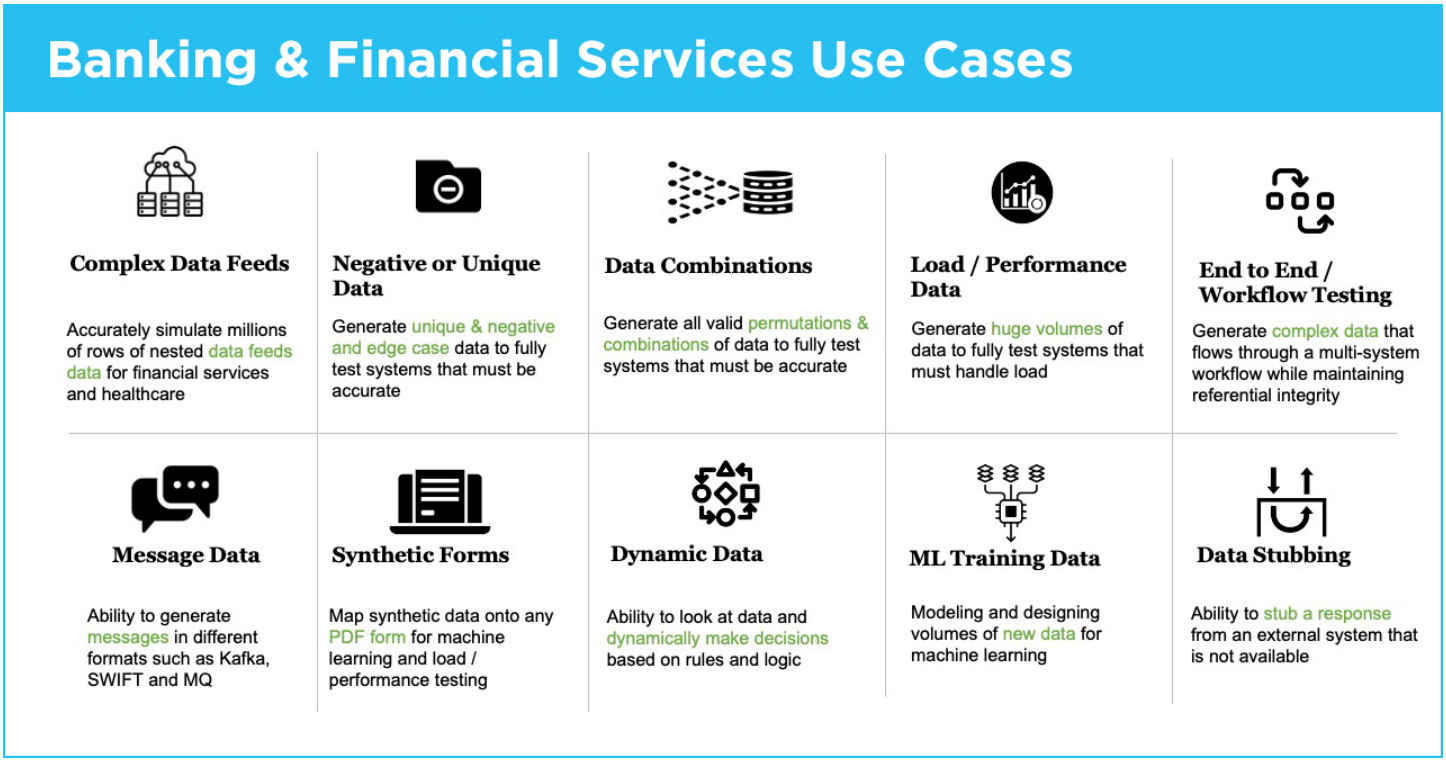
- Command line
- Batch file
- Shell scripts
- Scripting language
- Bash file (e.g., for Jenkins)
- Compiled language
- REST API
- Socket API
- GMUS (GenRocket's Multi-User Server)

These integration methods provide developers and testers with many options for integrating GenRocket with virtually any toolset for software development and automated testing.

Potential Synthetic Data Use Cases

GenRocket has the potential to address many more use cases at the bank. There is virtually no category of data that cannot be synthetically generated by the system. That includes all forms of structured data, unstructured data and even image data can be modeled and generated on-demand and in any volume.

In addition to all forms of relational database environments, GenRocket can easily generate big data and simulate NoSQL databases and IoT data streams as well. The chart below provides several use case examples that are typically found in the financial services sector for GenRocket's TDA platform.



GenRocket is also being used to simulate data feeds that represent complex workflows or event streaming platforms like Kafka. Here are some additional examples of the diverse synthetic test data types that can be generated by GenRocket.

- **Synthetic Bank Checks**
- **PDF Forms**
- **NACHA**
- **BAI2**
- **Blockchain Nexo**
- **Parquet**
- **Avro**

Many financial institutions that have deployed the GenRocket platform experience a dramatic improvement in operational efficiency and quality. The impact is typically realized in the form of reduced test cycle time and increased test coverage. Here are some examples of the benefits experienced by one GenRocket financial services customer across multiple value streams.

PAYMENTS

Cycle time reduction of 1400 hours and an increase in coverage for systems integration testing and performance from 30% to 80%.

BANK

Test cycle time reduction of more than 380 hours with an increase in regression and API performance coverage from 0% to 70%.

DATA & ANALYTICS

Reduced test cycle time by more than 200 hours with an increase in component testing coverage from 0% to 50%.

CARD

Saved over 1300 hours during the first 9 months of deployment with an increase in regression coverage from 0% to 50%.

GenRocket's **Test Data Automation** platform can have a similar impact on any software testing operation. The system is designed from the ground up to accelerate the provisioning of test data, increase code coverage and reduce the total cost of ownership compared to traditional **Test Data Management** systems.