International Journal of Multidisciplinary Research and Growth Evaluation.

Designing Rest Api Test Suites for Response Time Validation Using Jmeter and Postman in Retail Microservices

Udayan Verma
Denver, USA

* Corresponding Author: **Udayan Verma**

Article Info

ISSN (online): 2582-7138
Volume: 05
Issue: 06
November-December 2024
Received: 07-10-2024
Accepted: 07-11-2024
Published: 05-12-2024
Page No: 1886-1888

Abstract
We describe a domain-specific methodology for building test suites for REST APIs that measure response times for retail microservices using Postman for baseline latency checks and functional tests and Apache JMeter for load-driven performance measurement. In Charter Communications' retail systems, microservices are shotgunned for time-critical, order-intensive workflows, such as product catalogue queries, order submissions, and payments. Strict response time SLAs are critical to seamless customer experiences and minimising transaction abandonment rates. Postman functional validation is incorporated into CI pipelines for lightweight response time validation, and JMeter is utilised for sustained and concurrent load simulations reflecting real-world traffic conditions. This approach offers an integrated view of functional correctness and latency compliance; thus, regressions can be identified and rectified before performance affects production.

DOI: https://doi.org/10.54660/.IJMRGE.2024.5.6.1886-1888

Keywords: REST API, Performance Testing, Response Time Validation, JMeter, Postman, Retail Microservices

1. Introduction

In retail ecosystems, microservices support distributed transactions such as inventory checks and payment authorisation, where API response time is crucial for conversions and customer satisfaction. At Charter Communications, QA automation integrates Postman and JMeter. Postman validates functional correctness and baseline latency under minimal load, while JMeter performs load and stress testing, capturing SLA compliance and latency distributions under concurrent traffic. Such SLA thresholds as maintenance of the 95th percentile below 500ms raise an alert when violated. This framework, a part of CI/CD, has ensured high responsiveness in a catalogue search, complex checkout, and payment gateway integrations to provide high-performance retail microservices in a reliable and SLA-compliant manner.

2. Background and Rationale

The transition of monolithic to microservices structure in the retail industry has been necessitated by the fact that it is more agile, has a higher scale and is also more resilient. Nonetheless, this decentralised process comes up with new issues, especially where performance management is concerned. One bottleneck towards a full user journey can be represented by a single slow microservice. The business reasons behind a specific attention paid to response time validation can be justified, thus, with the business reality, which is a direct proportionality between poor business performance and slowness of response time [1]. The challenge presented in this report is solved with the help of a test plan that incorporates performance validation into the system.

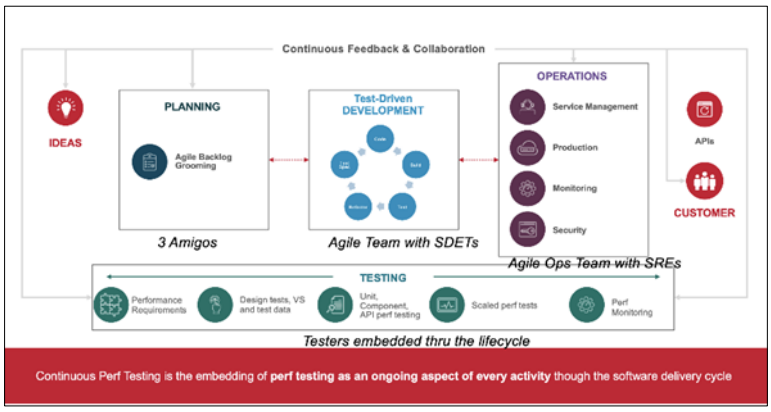

Fig 1: Hybrid Performance Testing Workflow for Retail Microservices

Postman and JMeter were chosen considering their complementary features that come up with a complete testing solution. Postman is a great performer in the preliminary stages of growth. It can be used to quickly construct functional tests, which may be augmented with straightforward performance assertions. Through such tests, a shift-left performance is accomplished by incorporating these tests in the CIC pipeline [2]. It means that it identifies issues of performance faster, which makes the solution of the problem less expensive and complicated.

The Postman, although it has the capability of testing the baseline, does not attempt to emulate real user charges like JMeter. JMeter informs of the behaviour of the system under stress conditions by chaining together customary utilisation patterns such as concurrency, constraining, and think period [3]. Response times and errors are the outcomes of a test, which prove whether the system can achieve SLAs in a production environment. Incorporating the two tools is a multi-layered protection against degradation of performance.

3. System Architecture & Pipeline Design

The architecture of the system can be combined with the existing CI/CD pipeline, during which automated validation of the system performance is performed at strategic levels of quality. Upon code being committed by a developer, a webhook causes the CI server to utilise it in building and deploying it to a test environment. Newman executes the Postman test pack, testing responses and functional correctness. Mistakes interrupt the construction, giving rapid feedback. In case of success, the code is tested through performance testing using JMeter to do resource-intensive tests [4]. These can be practised as individuals can plan a practice night before release, and those which are conducted on short notice. JMeter can work in a distributed fashion where a master node is touched and multiple slave nodes emulate thousands of concurrent users and prove realistic scalability. Outcomes are centralised and juxtaposed with SLA standards. In case of violation, automatic failure of the pipeline is triggered. The two-level nature of this approach assures every element of constant performance verification and keeps a code that fails to perform at an acceptable level.

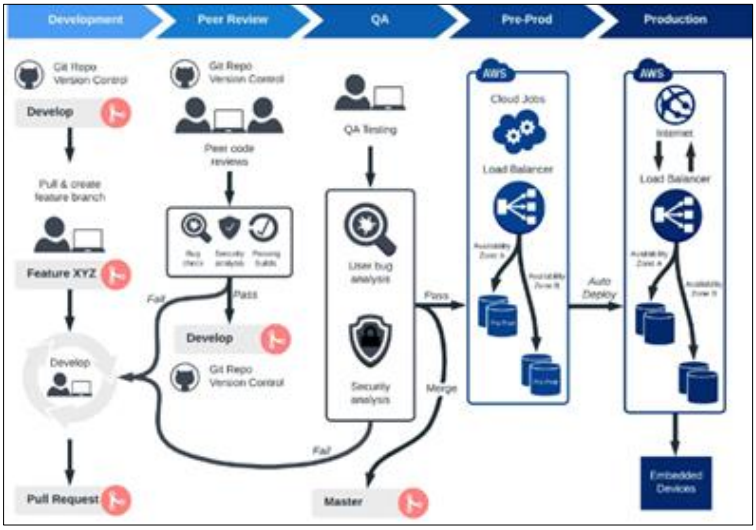

Fig 2: CI/CD Pipeline with Integrated Performance Quality Gates

4. Integration & Maintenance Strategy

The long-term viability of the testing framework must have a very clear integration and maintenance plan. All the test artefacts, such as Postman collections, the environment configuration and the JMeter test plans, are maintained in a version control system, along with the application. This is so as to make sure that the tests develop concurrently with the microservice that it is meant to test [5]. The CI pipeline is also set to constantly fetch the newest version of the tests to ensure that the most recent validation is done.

In order to make the test suites maintainable, a data-based modular approach is taken. In both Postman and JMeter, reusable test components are developed to prevent duplication of code and ease updating the code. The output of test data is externalised out of the scripts (test scripts) and maintained in other files or by a test data management service [6]. This isolation enables the test scenarios to be easily modified without the need to modify the test logic behind

them. An ongoing review mechanism is also established to make sure that the relevance of the suites of tests, as well as their validity, remains preserved, and a review of the test coverage, as well as the realism of loads, is provided.

5. Scalability & Optimisation

The testing framework is based on scalability, which will guarantee its expansion alongside the retail platform. The distributed nature of JMeter has provided it with the option of horizontal scaling, that is, it enables it to add load generator nodes, and this can be dynamically added by its cloud functions to provide cost-effective and flexible traffic simulation [7, 8]. As a performance, optimisation aims at efficiency: Postman tests are lean where critical paths are addressed to reduce the build time, and the JMeter test plan is based on parameterisation and realistic think times to avoid unrealistic stress. Moreover, it can generally be used together with application performance monitoring (APM) tools to further analyse the bottlenecks by identifying them in a short period. Combining the above guarantees that there is scalable, efficient, and insightful performance testing of retail micro services.

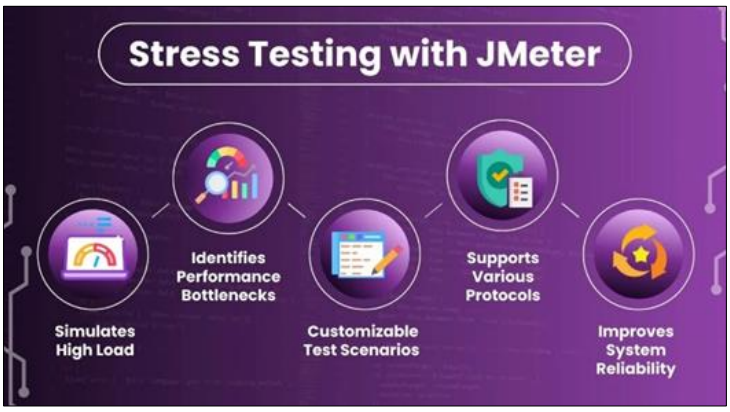

Fig 3: Stress Testing with JMeter: Techniques and Strategies

6. Reporting & Monitoring

Testing is maximised with effective reporting and monitoring. The results of Postman get incorporated into the CI / CD pipeline, which gives feedback on pass/fail and a brief summary. The detailed outputs of JMeter are filtered into a time-series database and demonstrated with such a tool as Grafana, which provides an interactive dashboard to track the trends and offload the performance problems. Financial performance management can be done effectively through constant improvement of these dashboards, which allows identification of discrepancies in expected behaviour, and this is automated. This makes sure that the investigation and resolution are obtained quickly before there is a problem with end users. The process of checking, monitoring, and development will create a causal authority and ensure reliability and will be the pillar of the development of a mature performance engineering.

7. Conclusion

In this report, a hybrid testing framework is proposed, a mixture of Postman and JMeter, to evaluate the response time of REST API in retail microservices. Making performance validation a part of the CI/CD pipeline is the way to ensure that the performance is consistently stable during development, so that the business risk associated with performance problems can also be minimised. This strategy is useful in providing cold, highly dependable, and scalable micro services that cater to SLA and provide high-quality customer experiences, which are essential to succeed in the current digital market environment.

Reference

1. Dragoni N, Giallorenzo S, Lluch-Lafuente A, Mazzara M, Montesi F, Mustafin R, *et al.* Microservices: Yesterday, today, and tomorrow. In: Mazzara M, Meyer B, editors. Present and ulterior software engineering. Cham: Springer; 2017. p. 195–216. https://doi.org/10.1007/978-3-319-67425-4_12
2. Postman. Integrate API tests with Postman, Newman, and Travis CI [Internet]. Postman Blog; 2017 Aug 23 [cited 2026 Apr 1]. Available from: https://blog.postman.com/integrate-api-tests-with-postman-newman-and-travis-ci/
3. Forrester Research. It's time for shift-left performance testing [Internet]. Forrester; 2019 Apr 19 [cited 2026 Apr 1]. Available from: https://www.forrester.com/report/its-time-for-shift-left-performance-testing/RES147361
4. Cao Q, Schniederjans DG, Schniederjans M. Establishing the use of cloud computing in supply chain management. Oper Manag Res. 2017;10(1-2):47–63. https://doi.org/10.1007/s12063-017-0123-6
5. Urošević V, Dagliati A, Ottaviano M, Vojičić N, Larizza C, Pala D. Design and optimization of REST services for performance and scalability in provision of big environmental data through an Earth observation platform. In: Proceedings of the IEEE International Conference on Consumer Electronics – Berlin (ICCE-Berlin 2022). Piscataway (NJ): IEEE; 2022.
6. Mahajan G, Attar V, Kalamkar S. Generation of JMeter scripts for performance testing of Moodle server. In: Proceedings of the 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N). Piscataway (NJ): IEEE; 2022.
7. Kumaresan KS. Performance testing as a service using cloud computing: A literature review and framework. Softw Pract Exp. 2022;52(12):2569–84. https://doi.org/10.1002/smr.2492
8. The Apache Software Foundation. Distributed testing step-by-step [Internet]. Apache JMeter User Manual. [cited 2026 Apr 1]. Available from: https://jmeter.apache.org/usermanual/jmeter_distributed_testing_step_by_step.html