ISSN: 2754-6683

Journal of Marketing & Supply Chain Management

SCIENTIFIC Research and Community

Review Article

Open Access

Canary Deployment in PEGA Platform: Enhancing Software Release Management

Aindrila Ghorai

Lead Architect, USA

ABSTRACT

Canary deployment is a progressive strategy for software releases, minimizing risks by gradually rolling out changes to a small subset of users before wider deployment. This paper investigates the implementation of canary deployment within the Pega Platform, a robust technology for Business Process Management (BPM) and Customer Relationship Management (CRM). It explores the benefits, challenges, and methodologies of canary deployment, presenting empirical evidence and best practices for its successful adoption in Pega environments. The research aims to provide comprehensive insights for practitioners and organizations seeking to enhance their release management processes and achieve smoother, less disruptive software updates.

***Corresponding author**

Aindrila Ghorai, Lead Architect, USA.

Received: June 06, 2022; **Accepted:** June 13, 2022; **Published:** June 20, 2022

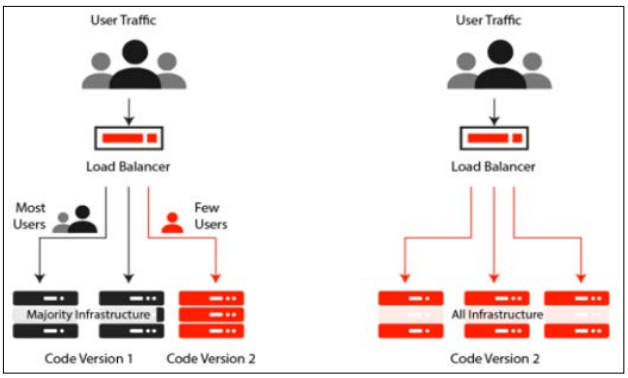
Keywords: PEGA, Canary Development, Release Management, Business Process Management, Customer Relationship Management

Introduction

The Pega Platform is widely recognized for its capabilities in automating business processes and delivering scalable applications. As organizations increasingly rely on Pega for mission-critical operations, the need for reliable and efficient deployment strategies becomes paramount. Canary deployment offers a controlled and systematic approach to releasing new features and updates, allowing teams to monitor and mitigate potential issues before a full-scale rollout.

Canary Deployment

Canary deployment is a strategy where new software versions are released to a small, representative subset of users before a wider rollout. This approach allows teams to observe the behavior of the new release in a real-world environment and address any issues that arise, reducing the risk of widespread disruption [1,2].



PEGA Platform

The Pega Platform is a comprehensive suite for BPM and CRM, offering tools for application development, business process automation, and customer engagement. It supports continuous integration and continuous deployment (CI/CD) practices, making it an ideal candidate for advanced deployment strategies like canary deployment.

Research Scope/Objective

This paper aims to:

- Analyze the principles and benefits of canary deployment.
- Investigate the feasibility and implementation strategies of canary deployment within the Pega Platform.
- Provide best practices and recommendations for organizations considering canary deployment in their Pega environments.

Technical Implementation

Business Objective

The business aims to implement iterative changes in the production environment as part of scheduled releases. To mitigate potential operational impacts, specific changes within the overall deployment package should initially be made available to a limited subset of users. This pilot approach will allow the business to assess the performance and stability of these changes. Upon successful validation by the pilot user group, these changes will be gradually rolled out to the broader user base.

Problem Statement

Canary deployment is a prevalent strategy in contemporary software technology stacks. Modern applications typically run code in containers, which fetch code from repositories that are version-controlled. This setup facilitates the implementation of canary deployments, allowing different repository versions to be targeted and gradually releasing changes from a small subset of

users to the entire user population.

However, with Pega, the scenario is different as the source code resides in the database. The code is served from the rules schema within the Pega database, presenting unique challenges for implementing canary deployments.

Solution Strategies

For the purpose of this paper, we will examine two features, feature 1 and Feature 2. The business objective is to deploy Feature 1 universally to all users, including pilot users. Conversely, feature 2 will initially be released exclusively to the pilot user group. Following successful validation and approval from the pilot users, feature 2 will then be incrementally deployed to the entire user base.

There can be a few methods to achieve this objective. We will analyze these options using the aforementioned deployment scenario involving two distinct feature sets.

Strategy #1: Leveraging Ruleset Version/Application Management feature

- Feature 1 is developed in application version 01.01.01.
- Feature 2 is developed in application version 01.01.02, ensuring it includes the rules from Feature
- A dedicated access group is created for pilot users that will point to version 01.01.02.
- Upon successful validation, the access group for the remaining users is updated to point to version 01.01.02.

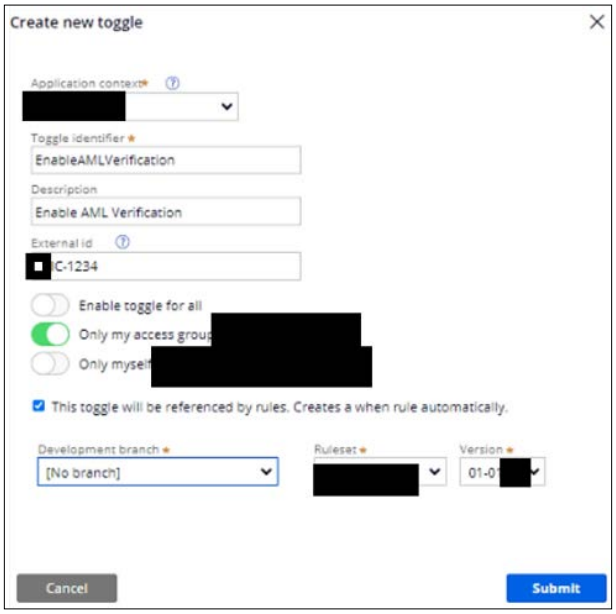
Consideration of Strategy #1 - Each subsequent change of this type must be developed in two separate application versions. In the event of Hazel cast synchronization issues between different machines, it may be necessary to downgrade an access group or point it to a higher application version. However, these changes might not take effect without a server restart to reset Hazel cast synchronization.

Strategy 2: Leveraging Toggle Management feature

This represents the most proximal mechanism providing canary deployment functionality out-of-the-box (OOTB) while requires minimal modifications post the initial setup. Subsequent sections in this paper will delve into the technical execution steps of this strategy.

Execution of Strategy #2

- The Toggle Management feature is accessible from the Dev Studio interface by navigating through Configure > System > Release > Toggles.
- To create a new toggle, users should click on the "Create new toggle" button. Subsequently, they need to input the toggle's name and designate the appropriate access group or user for whom the feature activation is intended.



- Upon clicking the "Submit" button, the Toggle feature element is generated.

- Aforementioned When rule is utilized within the code relevant to Feature Set 2, which requires validation by pilot users.
- Following the validation process, the feature can be globally enabled for all users.
- When the Toggle feature undergoes an update, allowing the selection between options such as "Enable toggle for all," "Only my access group," or "Only myself," a data instance is generated in the pr_data_toggle table within the data schema. During runtime, the associated When rule queries this entry within the data schema to verify if the setting is enabled. If enabled, the When rule evaluates to true, thereby exposing Feature Set 2 to the respective pilot user group. This method of runtime verification against a data instance directly minimizes the likelihood of encountering hazel cast synchronization issues.
- In scenarios where a feature set is executed in a background context, it is imperative to create a toggle feature for the access group utilized to execute the specific feature set.

Benefits of Canary Deployment

A/B Testing: Canary deployments facilitate A/B testing by allowing two versions of the application to be presented to users. This enables the collection of empirical data on user interactions and preferences, determining which version performs better.

Capacity Testing: Direct capacity testing in a large-scale production environment is often impractical. Canary deployments incrementally shift users to the new version, thereby uncovering performance issues progressively and providing insights into system capacity under real-world conditions.

User Feedback: Developers gain critical insights through feedback from actual users interacting with the canary release, allowing for informed improvements and adjustments based on real-world usage patterns.

Minimal Downtime: Unlike blue-green deployments, canary deployments ensure minimal downtime. In the event of an issue, the deployment can be swiftly rolled back to the previous stable version, maintaining continuous system availability [2].

Best Practices

Start Small: Begin with a very small percentage of users and gradually increase the traffic based on system stability and performance.

Rollback Strategy: Implement a proper rollback mechanism to swiftly revert to the previous stable version if issues are detected.

Continuous Monitoring: Use comprehensive monitoring tools to track performance metrics and user feedback continuously.

Stakeholder Communication: Keep all stakeholders informed about the deployment process and potential impacts.

Conclusion

Canary deployment within the Pega Platform offers a strategic advantage for organizations seeking to enhance their software release processes. By reducing risks, improving user satisfaction, and enabling data-driven decisions, canary deployment can significantly contribute to more reliable and efficient software updates. This research provides a foundational understanding and practical guidelines for implementing canary deployment in Pega environments, paving the way for smoother and more controlled software releases.

References

1. Sato D (2022) https://martinfowler.com/bliki/CanaryRelease.html, https://martinfowler.com/bliki/CanaryRelease.html.
2. Rastogi A (2022) Canary Deployment Strategy: Benefits, Constraints And How It Can Be Used For Application Traffic Management https://www.appviewx.com/blogs/canary-deployment-strategy-benefits-constraints-and-how-it-can-be-used-for-application-traffic-management/.