indium
Make Technology Work

Canarying Releases to Mitigate Real World Errors

A Whitepaper

Introduction To Software Releases

Downtime. Bugs. Angry users. Do these words ring a bell? You may want to reconsider your release management process if this is the case. In the fast-paced, ever-changing world of business IT, you can't afford to offer half-baked products. Nonetheless, 75% of issues are caused by changes in software or its environment.

To thrive, organizations must maintain effective agility to accommodate diverse consumer demands. This requires continuous software evolution throughout its lifecycle to stay current with the ever-changing market demands. That's where "Software Releases" come to play!

Software releases refer to updates to a software product or service. Different releases exist based on their purpose, scope, and user impact. These include patch releases, minor releases, major releases, feature releases, and beta releases. The frequency and combination of releases depend on an organization's software development and release lifecycle. Let's explore these:

The Software Development Lifecycle (SDLC) consists of six phases:

- **Phase 1 - Project Initiation:** In this phase, the project team is formed, the project scope is defined, initial requirements are gathered, and feasibility analysis is conducted.

- **Phase 2 - Requirement/Planning:** During this phase, detailed requirement gathering, analysis, and documentation take place. A plan for project execution is developed, including timelines, resource allocation, and risk management.

- **Phase 3 - Design:** In this phase, the software design is finalized, including architecture, interface design, and database design.
- **Phase 4 - Development:** During this phase, the software is actually coded and developed using various technologies and programming languages.
- **Phase 5 - Testing:** In this phase, quality assurance of the software is ensured through various testing strategies, including black box testing, white box testing, cross-testing, and regression testing.
- **Phase 6 - Release:** Finally, the software is deployed to end-users or customers after ensuring its quality and functionality through various testing methods.

Challenges

Having non-functional software can tarnish an organization's reputation and disappoint customers. Despite employing numerous testing strategies to ensure error-free software, achieving 100% accuracy is often impractical for various reasons, including:

Bugs and errors: Despite thorough testing, the software may have bugs and errors that have gone unnoticed or emerged post-release, affecting the user experience.

Compatibility issues: The software may not be compatible with certain hardware or operating systems, causing issues for some users.

Security vulnerabilities: Hackers may exploit vulnerabilities in the software, compromising data and security.

Scalability: As user demands grow, the software may face challenges in scaling to meet the increased load, leading to slow performance or downtime.

Customer support: Post-release, the support team may have to address user queries and issues, which can be time-consuming and resource-intensive.

Regulatory compliance: Software may have to comply with regulations and industry standards, which may require updates and modifications post-release.

The best strategy to mitigate the risks is with "Canary Testing."

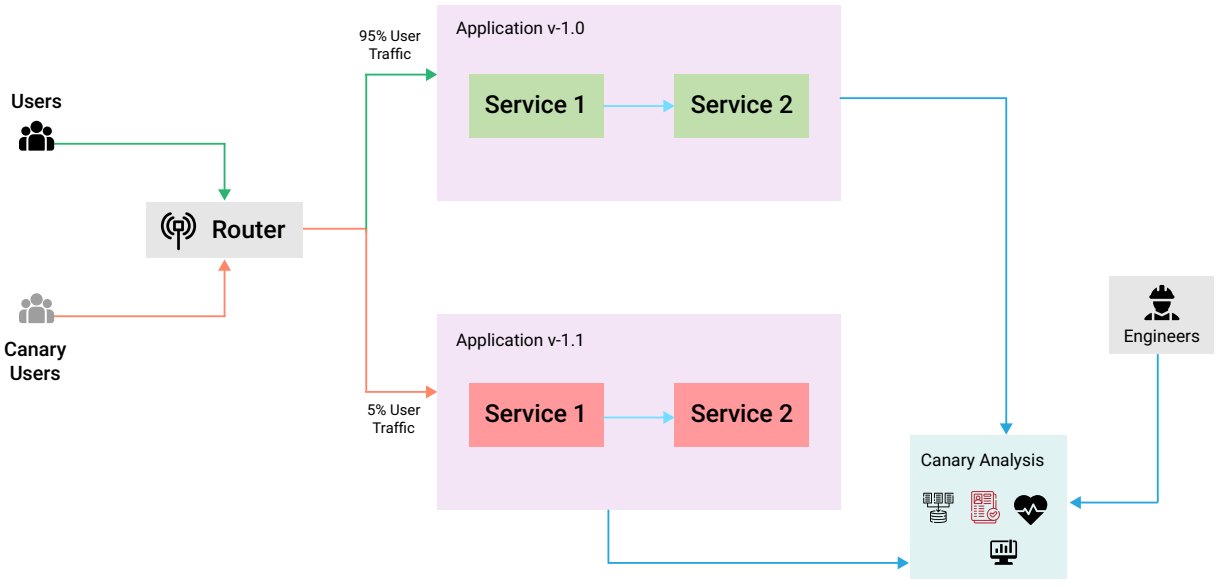
Canary Release = Canary Deployment

Canary testing is a method of testing new software features and functionality in production but with minimal user impact.

Instead of deploying the new release to all users at once, canary testing allows you to gradually roll out the update to a small group of users, also known as "canaries."

By monitoring the canaries' experiences, you can quickly identify any issues or bugs that may have been missed during testing.

Canary testing is also known as canary deployment or canary release.These terms are often used interchangeably, but they all refer to the same process of testing new software updates before deploying them to all users.



How to Perform a Canary Test? Planning and Implementing of Canary Test Deployment

Step 1: Select the Canary Users
Choose a small group of users, around 5-10% of the total user base. These users should be less impactful to the business and unaware that they are part of the testing group.

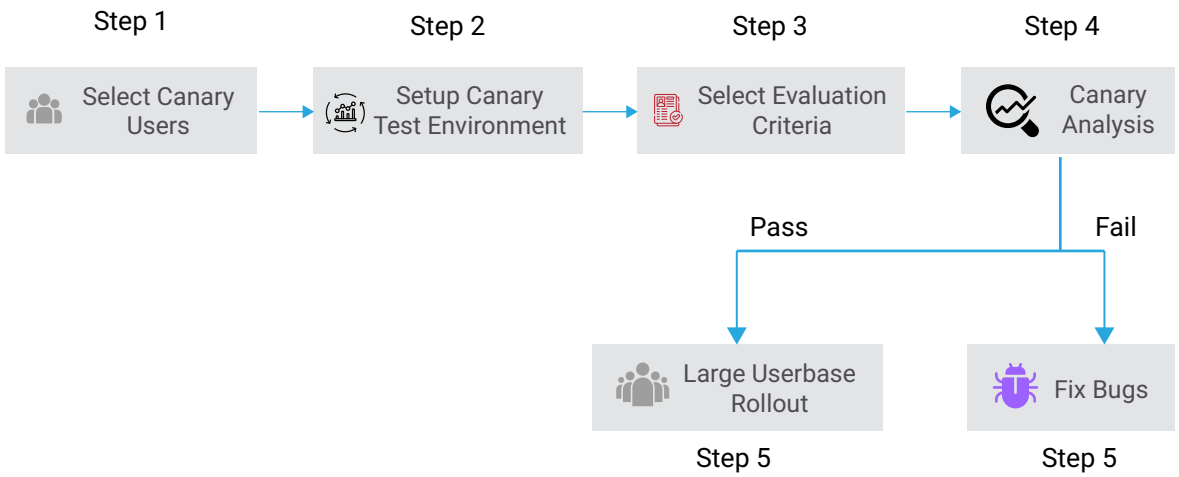
Step 2: Setup and Rollout Canary Test Environment
Set up the canary test environment with the new version of the software alongside the existing version of the software. Configure a router to route traffic between both environments and start the routing.

Step 3: Determine Evaluation Criteria and Time Frame
Set the evaluation criteria and the toolchains required to monitor the metrics. Decide on a time frame for the testers to analyze the metrics and conclude about the release's stability.

Step 4: Analyze the Release
Monitor and analyze the release for the decided time frame to see if it meets the evaluation criteria.

Step 5: Make a Decision
If the release meets the evaluation criteria, it can be rolled out to a larger set of users. If not, necessary bug fixes should be made before executing the major release to the larger group.

Complexities with Canary Testing



Like any other approach, Canary Testing has a few challenges. Here are some of the significant challenges of Canary Testing:

Mobile Testing: Although Canary Testing works wonders for web applications, it becomes a limitation for mobile applications since there is only one environment, the user's device, making it challenging to implement Canary testing.

Complex Manual Testing: When we release multiple features, testing them using multiple environments can become difficult and cumbersome. Upgrading and analyzing multiple environments manually can also add to the complexity of the entire process.

Bad User Experience: Some users may not want to be used as test subjects, resulting in a bad user experience. To be transparent, you can inform users that they are being used as "canaries" through an "early adoption" program or something similar.

Overcoming Complexities of Canary Testing

Feature Flags for Mobile Apps and Large Features

- Using a feature flag-based approach can help overcome the limitations of a single environment for mobile apps and the need for multiple environments for larger releases.
- Feature flags provide a controlled way to turn on changes or new features for specific subsets of your audience, as well as maintain operational toggles. They offer precise ad-hoc targeting around any dimension you want, allowing for more targeted testing and feedback.
- Enabling feature flags in the application can facilitate the execution of a canary test for a production release using a single production instance. It simplifies the process by toggling on or off a specific feature.

Automation Tests and Analysis Tool Chain for Effective Canary Testing

- Since canary testing involves at least two or more instances of the application, automating the entire process as much as possible is a wise choice. Automation testing tools make creating new tests easier, identifying test users, and analyzing test results. Engineering teams can also create predefined test cases to perform targeted testing and back up the test results.
- A comprehensive analytics platform can analyze different dimensions like errors, logs, performance metrics, and transactions, giving testers a holistic view of the application's performance. This helps to decide whether the release is stable or not.

Benefits

- With canary testing, real-world users can evaluate and analyze two software versions. Since the users are just a subset of total users, a release's impact would be much less.
- Adopting a feature flag-based approach can help minimize the cost of having two production environments.
- Canary deployment can also be used for A/B testing, as it offers two alternatives to the users and selects one with better stability.
- Since the newer version runs in parallel with the old version, rolling back would be straightforward.
- Since users for the canary testing are decided by us, we can select the users in such a way that we get the right feedback for the features.

Case & Point

Implementing Canary Release with Feature Flags for a Tax Computation Feature Update in HRMS SaaS Application

Client: Our client is the leading HR service provider in the US region.

Objective: To implement a canary release with a combination of feature flags to roll out a major change in the Tax Computation feature in the Compensation Module from version 1.0 to version 1.1 of the HRMS application.

Challenges: The introduction of new tax exemption sections in version 1.1 requires a change in declaring investments and affects the tax computation in the backend, potentially impacting the net salary of employees.

Any bug in the application would lead to their salaries being computed incorrectly, and for customers with a large employee base, this would create an adverse impact.

Solution: After analyzing what they are facing regarding this new version change and how it will affect their entire user base since this is a change in the application that our client is selling to their customers. We proposed a strategic canary plan with a team of experts that our client adopted to follow and execute a canary release:

Step 1: Select Canary Users

Our client selected trial customers as canary users, who may have a fallback mechanism and fewer employees on-boarded.

Step 2: Determine Evaluation Criteria and Time Frame

The following parameters were determined for the evaluation criteria:

- Transaction failures due to technical errors should be less than 2%
- Transaction failures due to logical errors should be less than 2%
- Average response time of API calls in the computation module should be less than 2 sec
- 50x errors in the API/Web Applications should be less than < 1%

The time frame to monitor the release was set at 24 hours post-release.

Step 3: Set up and Roll out Canary Test Environment

Feature flags were enabled for the compensation module, and the canary feature was enabled for the selected canary users through the admin console. An Event Management System was set up for parameters 1 and 2, and an infra monitoring system was set up for parameters 3 and 4 to obtain the statistics.

Step 4: Analyze the Release

The engineering team monitored the dashboard setup for the parameters and checked if any thresholds were crossed.

- Transaction failures because of technical errors were less than 2% → 1.0%
- Transaction failures because of logical errors were less than 2% → 1.2%
- The average response time of API calls in the computation module was less than 2 sec → 0.8 sec
- 50x errors in the API/Web Applications was less than 1% → 0.5%

Step 5: Decision Making

Since the SLO for the parameters was met, the engineering team concluded that the release was stable and enabled the feature for all customers.

Outcome: The canary release with feature flags helped ABC successfully roll out the tax computation feature update without major issues, ensuring minimal disruption to customers.

Point of View

A successful release is crucial for ensuring a positive user experience, which is essential for organizations to achieve customer satisfaction and success in the market. Implementing a reliable release process that minimizes the impact of bugs during large rollouts is crucial in achieving this.

While various release strategies are available, such as Blue Green Release and A/B Testing, Canary Testing is a widely used strategy that can effectively help organizations achieve their goals. Ultimately, the choice of strategy will depend on the specific needs of each organization, as the goal is to ensure a successful release.

USA
Cupertino | Princeton
Toll-free: +1-888-207-5969

INDIA
Chennai | Bengaluru | Mumbai | Hyderabad
Toll-free: 1800-123-1191

UK
London
Ph: +44 1420 300014

SINGAPORE
Singapore
Ph: +65 6812 7888

www.indiumsoftware.com

For Sales Inquiries
sales@indiumsoftware.com

For General Inquiries
info@indiumsoftware.com