Developing an Internal Notification Feature

CS4991 Capstone Report, 2024

Sofia Yang
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
syy6zn@virginia.edu

ABSTRACT

A major application hosting platform company needed a method for service teams to be able to receive notifications when a feature was failing. To address this problem, my team used AWS services, Java and React.js to develop a feature notifying customers when a given service is in a failing state. Implementing this solution will improve notification automation, because customers will no longer need to manually check for service statuses. This solution will also help to improve the detection and response times, because with the notification customers will be able to reduce their mean time to detect (MTTD). In the future, there will be other features, such as improving the scalability of the notification feature to allow multiple customers to receive notifications and allowing the notification to receive follow ups upon the recovery of the service.

1. INTRODUCTION

This summer I worked on an internal team responsible for ensuring that on launch day services will be ready for customers to use. In order to achieve this goal, the team has a service that aggregates test cases into test suites, that can be customized by the customer, defining requirements that a given service must satisfy to be considered ready for launch.

Currently, customers can only manually check the front end console to identify the status of a service. This is a problem because it could lead to delays in launching the servers to customers and longer mean time to detect (MTTD), the metric that measures the amount of time it takes to be aware of a problem with the service.

2. RELATED WORKS

In order to implement the notification feature, I read through the existing codebase for both the backend and frontend, to determine how to integrate my feature into the existing service. The implementation of this feature involved the integration of an internal API, in order to understand the process of integration this API, I read through the onboarding instructions provided by the internal team that owned this API. In addition, I read through the implementation examples provided by the API owner team and examples of API integration done by other teams.

The new-integrated feature will send notifications that will be easily viewable by the company set up on-call queues. The decision to send notifications to on-call queues was influenced by the benefits of reducing detection time by sending the notification through the on-call queue, as outlined by Augustian (2022).

In addition, I read the articles, documentations and developer guides on the *AWS Documentation* (AWS, 2019) for the AWS tools used for the project to understand how they can be applied to the project implementation

3. PROJECT DESIGN

This project consisted of three main components. 1) allowing the database to store whether or not the test suite has enabled the notification feature and where the notification should be sent. 2) enabling the backend to determine when the notification should be sent and sending the notification to the customer and 3) developing the frontend so customers will be able to enable the notification feature.

3.1 Database and Backend Implementation

The first component was to update the existing database with the new information to support the notification feature. The second, the backend Java codebase was updated to send the customer the notification when the test suite changed from a pass to fail status. Once checked and after making sure the test suite has not been notified of a status change in the past, the backend will trigger a Lambda (AWS Service) using the internal API to send the notification to the customer chosen on call queue.

3.2 Frontend Implementation

The third component is to update the front end console so customers can enable the notification feature on their desired test suite and choose the on-call queue they want the notification to be sent to. To implement this, I updated the frontend React.js codebase to create a dropdown allowing customers to enable the notification feature, update the on call queue they want the notification to be sent or disable the notification feature for the test suite (Figure 1).

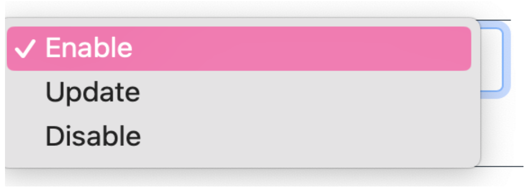

Figure 1: Dropdown for customers to enable, update or disable the notification feature

Once the customer chooses to enable the notification feature, they will be able to select their desired on call queue to send the notification to. When selecting update, the customer will be able to change the on call queue they would like future notifications sent to. When selecting disable, the customer will no longer receive any notifications about status changes for the test suite.

3.3 Testing and Code Review

For both the backend and frontend development, there were smaller milestones for the task. Each milestone has exhaustive unit tests to ensure the functionalities. To check the end-to-end function, both the backend and frontend had their own set of integration tests. For the backend, the integration tests ensure notifications would be sent under the right conditions. For the frontend, the integration tests ensure the customers will view the correct information when trying to enable, update or disable the notification feature.

To ensure the quality and functionality of the code and test cases, every milestone has its own code review, reviewed and revised until at least two software engineers have approved it. Once approved, the code will be published to the codebase and moved through the different stage (beta, gamma and production) in the pipeline.

4. EXPECTED OUTCOMES

This project had not been officially announced to customers at the end of my internship. However, with the notification feature implemented, it is expected that service teams will be able to reduce the amount of time it takes to be aware of a problem in their service. In addition, this will also reduce the frequency of customers noticing the failures of the service.

With the notification feature implemented, it is anticipated that customers will enable notifications for their test suites. Once enabled, customers will be able to receive the notification in their on-call queue and respond to the failure as needed.

In addition, in the implementation of this project, the internal API that was integrated has other endpoints that can be used by the team for other planned projects.

5. CONCLUSION

In conclusion, in the process of this working on this project, I learned how to use various AWS tools and how to work with frontend tools. I was also able to improve my skills in writing more reusable and maintainable code.

6. FUTURE WORK

Going forward, the team would like to add direct access to on-call notification, so customers can easily view and check the status of the problem resolving. Another feature the team would like to add is allowing multiple on-call queues to receive the notification about the failing status of a test suite. Currently, the only way for multiple customers to be aware of a fail status is through creating a separate copy of the test suite and enabling their on-call queue to receive the notification, which is not as customer-friendly as we would like.

REFERENCES

Augustian, T. (2022, January 25). *Embracing on-call system in software development process*. Medium. Retrieved 28, November 2023 from https://medium.com/tokopedia-engineering/embracing-on-call-system-in-software-development-process-c74b030e6565

Amazon Web Services (2019). Amazon.com. Retrieved 28, November 2023 from https://docs.aws.amazon.com/