Module 23 – Amazon EventBridge – Part 3

Overview

This is the last EventBridge-related module. You will complete what you have done in the previous two modules by configuring a new rule that forwards events to a lambda (the parts of Figure 1 surrounded by a red-dashed rectangle).

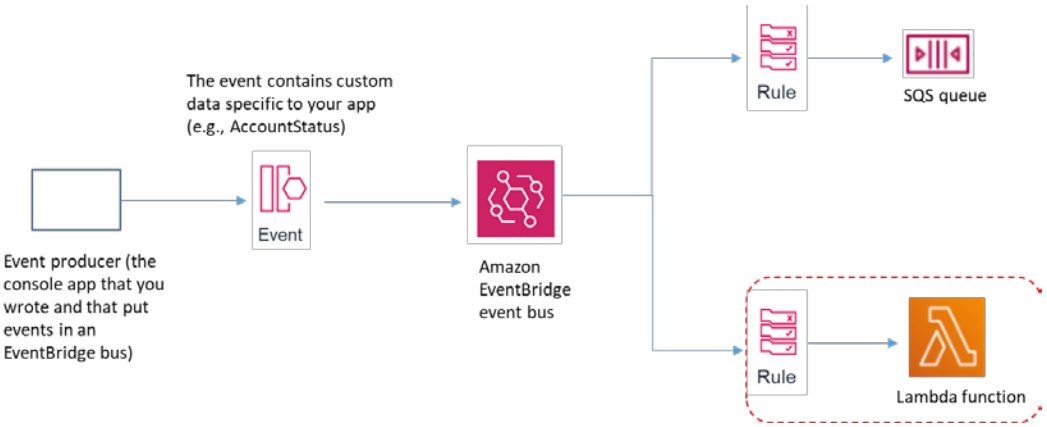

Figure 1: Adding additional rules and targets to an EventBridge bus.

In the previous module you already did the upper part of Figure 1 (a rule that forwards events to a queue when *AccountStatus=Locked*). In this module you can use a different AccountStatus value in the lower rule (e.g., *AccountStatus=Deleted*) to end up with a system that behaves as shown in Figure 2. For the Lambda implementation, just print the event to CloudWatch. This demonstrates that you got the event and you can do any custom logic you want.

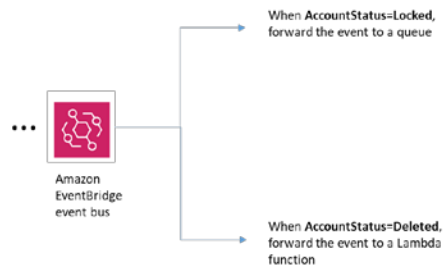

Figure 2: Route events to different services based on some value specific to my application.

You should now see and appreciate the benefits of EventBridge. We can summarize them as follows:

1. **Decoupling and separation of concern**:

   The producer of events (your console app) is decoupled from the components that consume the event data (the queue and the lambda). Consumers can be changed, updated, enhanced, or even completed removed without perturbing the producer. The producer now simply does not care about how many consumers consume the events it is emitting

2. **Parallelism and Developers Productivity**:

   Systems that are tightly coupled suffer from teams waiting on dependencies that need to be completed by other teams. This causes delays and loss of productivity. When you decouple a system, teams can work on different parts in parallel without waiting for each other, and therefore increases engineers' productivity and project completion time (Figure 3).

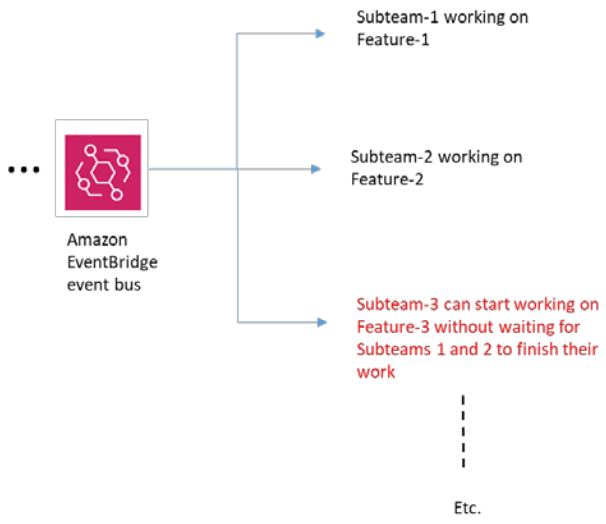

Figure 3: Decoupling improves work productivity by eliminating dependencies

What to Submit

Nothing to submit for this module.