Unit Tests

I'll make sure each method or function within every microservice has its own unit tests. These tests would focus on ensuring the correctness of self-contained units within the service. Any dependencies like databases or external services would be mocked.

Integration Tests

While unit tests cover isolated behavior, I want to confirm that my services are interacting correctly with their dependencies, such as AWS S3 or RabbitMQ. I'll set up integration tests that run against mocked versions of these dependencies.

Component Tests

I would write component tests that would examine the interactions between my services. For these tests, each service would be treated as a black box where I'm primarily concerned with the correctness of inputs and outputs.

Contract Tests

A contract test would ensure that the services are able to interact correctly with each other. This helps to identify any breaking changes in the service APIs. I would use tools like Pact or Spring Cloud Contract for this.

End-to-End Tests

Finally, I would set up end-to-end tests that run against the entire system to validate real user scenarios from start to finish. These tests would run less frequently due to their complexity and running time. In terms of efforts, I would mostly invest in unit tests, followed by integration tests, and then equally distribute my effort between component tests, contract tests, and end-to-end tests. This can vary depending on the specifics of the project.