Title: Software Testing in Distributed Systems: Apache Kafka

Abstract:

During the semester, we have explored different aspects of software testing on a monolithic software application but how does this apply to the highly distributed systems of today’s computing platforms. We aimed to explore this broad field by focusing on how to take lessons learned in class and apply them to our professional workplace in distributed systems.

As the authors, we have worked in distributed systems with both of our employers using cloud computing platforms and furthermore, distributed systems in production environments. However, either one of our employers are utilizing a testing framework to confirm proper behavior of these systems. Why?

Testing in distributed systems is hard and requires specialized skills and resources to do it in repeatable, and reliable fashion. For example, testing in a monolithic software application has a very general testing pattern of instantiate class/method/object, pass in parameters and validate. However, this pattern doesn’t work in distributed systems because a key principle that define what a distributed system is redundancy and failover. In order to achieve redundancy and failover, a distributed application or system must be deployed in a cluster. This means when we pass in a value, any node in the cluster could act on the value which means we have to validate the cluster as a whole since we have no insights on which node or application is executing on that value.

This is a huge problem but for the purpose of this paper we are going to prove out some simple testing principles by building a few simple tests to provide example of how this can be achieved. The tests we are writing are to validate proper behavior for a very common distributed messaging broker called Apache Kafka.

Keywords:

Distributed Systems, Software Testing, Apache Kafka

Introduction - Dale

Distributed System Challenges

Problem we are solving?

Approaches

1. DuckTape written by Confluent
2. AWS Virtual Machines
3. Docker/Jenkins/AKS

Technique - Dale

What is Apache Kafka?

Kafka® is used for building real-time data pipelines and streaming apps. It is horizontally scalable, fault-tolerant, wicked fast, and runs in production in thousands of companies.



What is Apache Zookeeper?

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications. Each time they are implemented there is a lot of work that goes into fixing the bugs and race conditions that are inevitable. Because of the difficulty of implementing these kinds of services, applications initially usually skimp on them ,which make them brittle in the presence of change and difficult to manage. Even when done correctly, different implementations of these services lead to management complexity when the applications are deployed.

What is Amazon Web Services?

Amazon Web Services (AWS) is a secure [cloud](https://aws.amazon.com/what-is-cloud-computing/) services platform, offering compute power, database storage, content delivery and other functionality to help businesses scale and grow.

What we built?

Kafka on AWS:

Kafka Producer

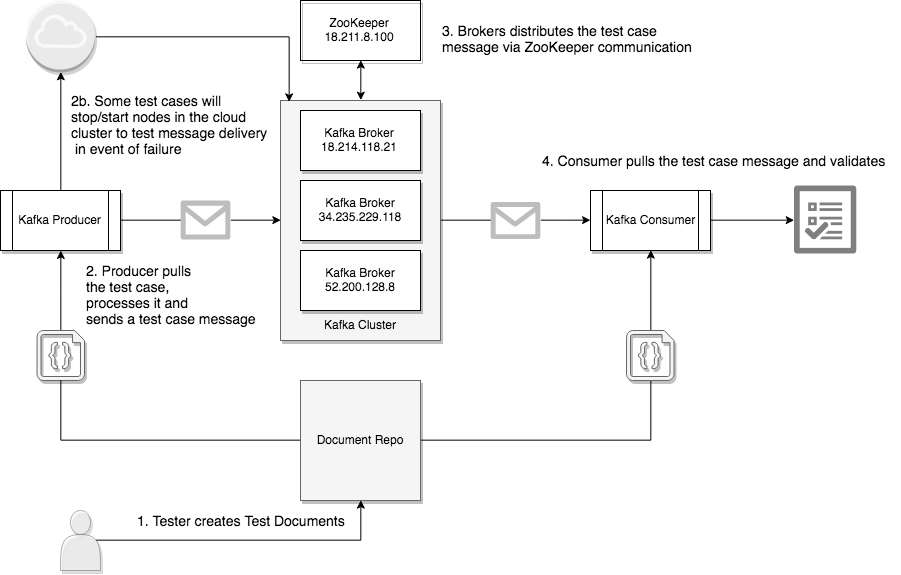
Kafka Consumer

Troubleshooting Issues

Running Kafka as Daemon

Java Heap Space

Illustration - Dale



Evaluation - Howie

How well did this work?

Improvements?

Related work - Howie

Confluent

Research Papers?

Conclusion - Howie

References

<https://kafka.apache.org/intro.html>

<https://zookeeper.apache.org/>

<https://www.confluent.io/>