Testing Tech Talk Proposal

Moti Begna, Kalven Schraut, Rohit Bagda

Topic: **Microservice Testing - What's easy? What's hard? And what's necessary?**

Microservices are widely used as a software architecture pattern in modern SAAS organizations. They help modularize, scale, and reuse components and services. We want to explore the different challenges of testing microservices in small as well as large sized organizations. We will also look into exploring testing across different microservice architectures and what different tools help us test microservices and their impact on the software development lifecycle.

Some (but not all) the ideas we were thinking about exploring in the tech talk:

- Different kinds of microservices?
- Different kinds of tests/ testing
    - Unit/Integration/Automation Tests
    - System Tests
    - Acceptance Tests
    - End to End Tests
    - Scale Testing
    - Stress Test
    - Fault Tolerance Testing
    - Manual QA Testing
- Challenges with testing
    - Contractual changes
    - Mocking behavior for cloud components.
    - Mocking behavior for upstream and downstream services.
    - Updating tests when making breaking changes to upstream services
    - CI Testing setup.
    - Costs of running tests
        - Running tests in an integrated environment can produce artifacts that can break other services.
        - Running thousands of tests in a non-integrated ephemeral environment can be expensive.
- Tools that can help us testing microservices
    - Cloud Emulator - Localstack, GCloud beta emulators.
    - Docker Containers.
    - Github workflows
    - Spring MVC Test Framework

Some of the resources we'll use in our investigations will be our own personal experiences with microservice testing at work, using our knowledge of some of the benefits and pitfalls of dealing with testing microservices. There are also many examples of open-source microservice applications on github that we can look at in order to explore different testing frameworks, getting a better understanding of why one framework might be used over another.