Path to Microservices

Rajadurai Krishnamoorthy
Principal, Software Architect
Monolithic Architecture

- Application is too large and complex
- Application start up is very slow
- Deploy the full application every time
- Continuous deployment is difficult
- Barrier to adopting new technologies
- Difficult to scale
Microservices Architecture

- Single Responsibility Principle
- Explicitly Published Interface
- Smart Endpoints & Dumb Pipes
- Independent DURS
From Monolithic to Microservices

1. Inter Service Communication
2. Distributed logging
3. Transaction Spanning
4. Finding root cause
5. Cyclic dependencies between services
6. Health Monitoring
7. Testing can become challenging
INTER SERVICES COMMUNICATIONS

- API Versioning
- Session Management
- Service Registry
- Transition Failures
- Rollbacks
Distributed Logging

- Own Logging Mechanism
- Many GBs of Unstructured Data
- Tracing and Profiling
- Zipkin/Slunk/ELK/Spring sleuth
Transaction Spanning
Distributed transactions
Finding Root Cause

- Problematic service
- Pull out all the data – Stack trace, variable values
- Data about unusually high processing times
- Deploy the change with more loggers
- Centralized root cause detection tool
CYCLIC DEPENDENCIES BETWEEN SERVICES

Vulnerable to distributed stack overflow errors

Certain transaction might be stuck in loop
Health Monitoring

What to monitor?

Different technologies

Live on different machine/container

Centralized logging and monitoring

Requires bigger investment and planning
Testing Can Become Challenging

- Individual microservice testing
- Dynamic environment
- Indistinct behaviors from the microservices interactions
- Forward / Backward compatibility testing
GREAT INDIAN DEVELOPER SUMMIT 2019
Conference: April 23-26, Bangalore

Register early and get the best discounts!

www.developersummit.com
@greatindiandev
bit.ly/gidslinkedin

facebook.com/gids19
bit.ly/saltmarchyoutube
flickr.com/photos/saltmarch/