

Building a Distributed Platform

Frank P Moley III

Titanium Sponsors

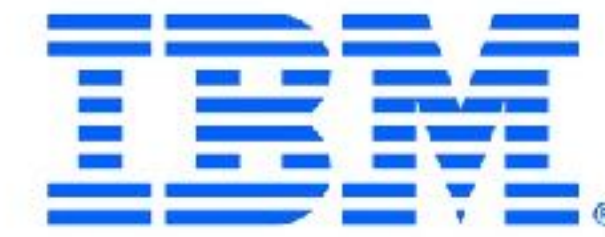


World Wide Technology



Couchbase

okta



Platinum Sponsors



NIPR
NATIONAL INSURANCE
PRODUCER REGISTRY

jack henry
& ASSOCIATES INC.



Abbott

K KOCH



Veterans United.
Home Loans

Gold Sponsors



service
management
group



COX AUTOMOTIVE



Agenda

- * Needs solved by a Distributed Platform
- * Components of a Distributed Platform
- * Tradeoffs and Decisions
- * Lessons Learned

Who I am

- * Lead Engineer for DataStax
- * Cloud Native developer passionate about architecture, security, and DevOps
- * LinkedIn Learning Content Author
- * @fpmoles

A distributed platform is a runtime for cloud native applications that spans multiple datacenter locations, zones, regions, or public and/or private providers.

Needs solved by a
Distributed Platform

Easy to Use

- * Most devs don't care about how their apps run
- * Few devs have the bandwidth understand everything involved in a runtime and their problem domain
- * Few devs want to be involved in building and deploying applications

Simplicity

- * Devs just want their apps to run
- * Devs just want customers using their apps
- * Customers want apps to be responsive

Responsive

- * Devs needs to respond to testing
- * Devs need to respond to production issues
- * Devs need to be alerted of production issues
- * Devs need to improve performance

Components of a Distributed System

Runtime

- * Bare Metal
- * VMs/Cloud VMs
- * Containers (Kubernetes, Managed Kubernetes, Cloud Foundry)

Onboarding

- * SCM
- * Build System
- * Deployment System
- * Static Code Analysis System
- * Unit Testing System
- * Integration Testing System
- * Acceptance Testing System

Monitoring

- * Metrics
- * Log Aggregation
- * Tracing
- * Alerting/Paging

Data

- * Core Distributed Need - Eventual Consistency
- * Database selection is critical -> Active/Active
- * Privacy and Security Considerations
- * Data Remediation

Ancillary Systems

- * Secrets Management
- * Infrastructure Management
- * Communications Management
- * Remediation Management
- * Authentication Management

Tradeoffs and Decisions

Lessons Learned

Managing Tech Debt

- * Put security in place first
- * Fail fast, and pivot when needed
- * Don't chase the hotness
- * Keep moving parts to a minimum

Managing Dev Expectations

- * Document early and often
- * Solicit feedback from your stakeholders
- * Don't become an order taker
- * Anticipate needs and deliver before the need arises
- * Over deliver

Instill Confidence Early

- * Alarm early and in lower environments
- * Establish dashboards and status pages quickly
- * Communicate deployments of the platform
- * Don't miss deadlines

Trust the Vision

- * Everyone will have an opinion
- * Listen and offer suggestions
- * Successes will attack believers