Overview

Team & Business

* Separate domain
* Separate team
* Take ownership
  + Developer-driven tools

Infrastructure

* Separate envs (QA/DEV/DEMO)
* Dedicated integration environment (how often to integrate)
* Same env config
* Automation
  + Automated DB scripts (liquibase)
  + Automated / semi-automated migration (need to plan it!)
  + Automated deployment (one-button)
    - CI: Check
      * jenkins
    - CI: Release
      * jenkins
    - CI: Deploy
      * Jenkins + chef
    - CI: Accept (automated acceptance tests)
      * Jenkins + selenium
    - Chef
* Monitoring

Design

Data

* Polyglot persistence
* No direct access to storage

Interaction

* Protocols
  + Documentation (must have)
  + Standardization (what protocols and formats to use)
    - REST/JMS with JSON as format
  + Client Libs
* Sync
  + As fast as possible (2-3 secs)
  + Requires user attention
  + Much caching (guava cache + JMX toolset)
  + REST
* Async
  + Scheme
  + JMS (activemq + camel)
* Failures
  + be ready for failures
  + be ready to recover
  + find where can fail
  + decide how to recover
    - Sync
    - Async
      * Automated
        + Redelivery (try one more time -> camel)
        + Exponential backoff
        + Require attention only last failure
        + Move to DLQ
      * Manual
        + You need to be notified (support tickets?)
* Logging
  + Enough logs
  + What to log (essential context, app id, user)
  + For interaction failures: all the nonsensitive data
  + Same format for all (shared config with logback include use)
  + Correlation ID
  + Centralized logging (Splunk, ELK)
    - With email sending (ownership!)
* When to use microservices