Microservices: The Big Picture

Microservices:

* Does one thing
* Bounded Context
* Scope of functionality
* Subdomain specific
* Autonomous
* Lightweight Communications (HTTP, Messaging)

Problems Monolithic Design:

* Productivity inversely proportional to size of Application
* Emerging tech cannot be used
* Individual components cannot be scaled

Microservices patterns and practices:

* Sharing DB is discouraged
* One team per subdomain
* Versioning becomes important when microservice interact with one another
* Eventual consistency no immediate consistency
* Aggregation of user interfaces
  + Server side aggregation
  + Client side aggregation
* Communications
  + RPC (Synchronous)
  + Messaging (Asynchronous) - Messaging Brokers e.g. kafka
* Microservices expose API using contracts e.g. SOAP, REST etc
* API contracts may differ depending on type of devices
  + Desktop (Good Internet Connection; more data)
  + Mobil (Poor Internet Connection: less data)
* Distributed Services
  + Service Registry - phonebook of services e.g. Eureka, Zookeeper
    - Self-Register - Start
    - De-Register - Stop
* Circuit Breaker e.g. Hysteric, JRugger
  + To avoid domino effect i.e. failure of one service leads to the failure of others
* API Gateway:
  + Identity and Access Management
  + UI Aggregation
* Availability
  + Avoid Single Point of Failure (SPOF)
    - Gateway
    - Message Broker
    - Service Registry
    - IAM
* Monitoring
  + Centralized
  + Dashboard
* Health check API
* Log Aggregation
* Metrics\Auditing
  + Record User Activity
* Rate Limiting
  + Denial of Service attacks
* Alerting
* Distributed Tracing
* Containers e.g. Dockers
* Orchestrators e.g. Kubernetes
* Continuous Delivery
* Environment (Dev\QE\Staging)
* Chaos Monkey