Notes

Lesson 8

- **Advantages and disadvantages of SOA**
  - Advantages
    - Independent services
    - Seperations of business processes and service lceogic.
    - Architecture is optimized for the business.
    - Reuse of services
    - Architecture flexibility
  - Disadvantages
    - Complex ESB (Enterprise Service Bus)
    - Changing the business process while still business processes are running is difficult
    - Most SOA are built on top of monoliths
- **What are the characteristics of monolith**
  - Everything is implemented in one big system
  - Can evolve into a big ball of mud
  - Hard to understand and change
  - Limited reusability
  - All or nothing scaling
  - Single development stack
  - Doesn't support small agile teams
  - Deploying monolith taks alot of ceremony
  - Deploying is high risk
  - Cannot be deployed frequently
  - Long build-test-release cycles
- **What are the characteristics of microservices**
  - Simple and lightweight
  - Runs in an independent process
  - Language agnostic
  - Decoupled
- **Microservices Boundaries**
  - DDD bounded context
  - Autonomous functions
  - Size of deployable unit
  - Polyglot Architecture
  - Selective Scaling
  - Small agile teams
  - Single responsibilities
  - Replicability and Changeability
  - Coupling and Cohesion
- **Orchasteration vs Chereography**
  - Orchasteration is one central brain
    - Advantages

            - Full control of the synchronous flow
            - Easy to monitor the process
        - Disadvantages
            - Coupling
            - Orchestrator is a single point of failure
            - No parallel processing
    - Chereography is no central brain
        - Advantages
            - Less Coupling
            - Fast: Parallel Processing
            - No single point of failure
        - Disadvantages
            - Hard to monitor the process
- **Stateless vs Stateful**
    - Stateful
        - The service contains in-memory state
        - State is maintained between requests
        - Advantages
            - Fast
        - Disadvantages
            - Synchronization issues
            - Hard to scale
    - Stateless
        - No in-memory data
        - All data is stored outside the service
        - Advantages
            - Database takes care of syncronization
            - Easy to scale
        - Disadvantages
            - Performance issues
    - Always prefer stateless over stateful
- **Microservices Deployment**
    - Multiple service instance per host
        - Advantages
            - Efficient resource utilization
            - Fast deployment
        - Disadvantages
            - Poor isolation
            - Poor visibility of resource utilization
            - Difficult to constrain resource utilization
            - Risk of dependency version conflicts
            - Poor encapsulation of implementation technology
    - Service per VM
        - Advantages
            - Great isolation
            - Great manageability

            - VM incapsulates implementation technology
            - Leverage cloud infrastructure for autoscaling/load balancing
        - Disadvantages
            - Less efficient resource utilization
            - Slow deployment
    - Service per container
        - Advantages
            - Great isolation
            - Great manageability
            - container incapsulates implementation technology
            - efficient resource utilization
            - fast deployment
        - Disadvantages
            - Technology is not mature as VM
            - Container is not as secure as VM

Lesson 9

- Why service registry discovery
    - Loosely coupled services
    - Increase application resiliance
- Ribbon is used for round robin.

Lesson 10

- Zuul
    - Cross Cutting Concerns
        - Security, Logging, Tracking, Transformation
        - Implemented with filters
            - Pre-filters
            - Post-filters
            - Route-filters
- Zipkin
    - Distributed Tracing
        - One central place where one can see end-to-end tracing of all communication between services
    - Spring cloud sleuth
        - Adds a unique Id to request so we can trace it
        - Also embeds these unique ids to log messages
- Elk Distributed Logging
    - When we need to collect all logs from all running services
- Reseliance
    - Fallacies of distributed computing
        - The network is reliable
        - Latency is zero
        - Bandwidth is infinite
        - The network is secure

        - Topology doesn't change
        - There is one administrator
        - Transport cost is zero
        - The network is homogeneous
    - Reseliance patterns
        - Timeouts
        - Circuit Breakers
        - Bulkheads
    - Timeouts
        - put timeouts on all out-of-process calls
        - log when timeouts occur
    - Circuit breaker
        - Fail gracefully
            - Fallback to alternative
        - Fail fast
            - Return an immediate response
        - Recover Seamlessly
            - Half open cuircuit check periodically if the service is up.

Lesson 11

- **Command Query Responsibility Segregation**
    - Seperates the querying from command processing by providing two models instead of one.
        - One model is built to handle and process commands.
        - One model is built for presentation needs (queries).
- **Problems for having one model for both querying and command**
    - To support complex views and reporting
        - Required domain model becomes complex
        - Internal state needs to be exposed
        - Aggregates are merged to view requirements
        - Repositories often contain many extra methods to support presentation needs such as paging, and free text search
    - Result: single model that is full of compromises.
- **How CQRS is done?**
    - Domain model is used for commands
    - View model is used for queries
- **Properties of Doman and view models**

	Command	**Query**
Consistency	Needs Consistency	Eventual Consistency is mostly Ok
Data Storage	You want normalized 3^{rd} normal form	Denormalized (1^{st} normal form is good for performance (No joins))

	Command	Query
Scalability	Commands don't happen very often, scalability is ofen not important	Queries happen very often, scalability is important

- **Views will become eventually consistent**
- **Reactive Streams are non-blocking**
- **Difference between Mono and Flux**
    - Mono<T> for handling 0 or 1 element
    - Flux<T> for handling N elements
    - You can subscribe to a Mono or Flux
        - Run some code when an object arrives in the Mono or Flux
- **Advantage and Disadvantage of Reactive Systems**
    - Advantages
        - Performance
            - No oneed to wait till all results are available
        - Scaling
            - Less threads needed
    - Disadvantages
        - The whole calling stack needs to be reactive
        - Harder to debug
- **Transactions**
    - A **Transaction** is a unit of work that is:
        - **Atomic**: The transactions is considered a single unit of work. either the entire transaction completes or the entire transaction fails.
        - **Consistent**: A transaction transforms the database from one consistent state to another consistent state.
        - **Isolated**: Data inside a transaction can not be changed by another concurrent process until the transaction has been committed.
        - **Durable**: Once committed, the changes made by a transaction are persistent.
- **Local transactiones**
    - The transaction is managed by the database
        - Simple
        - Fast
    - Always try to keep transaction boundaries within a service
- **Advantages and Disadvantages of Saga with commands (orchestration)**
    - Advantages
        - More control over transaction
        - Easier to monitor transaction
        - No cyclic dependencies
    - Disadvantages
        - Needs an extra orchestrator service
        - Orchestrators become complex too.
- **Advantages and Disadvantages of Saga with events (chereography)**
    - Advantages
        - Simple
        - Loosely coupled services

    - Disadvantages
        - Difficult to track who listen to which events
        - Possibility of cyclic events

Lesson 12

- **Aspects of Security**
    - **Authentication**: are you who you say you are?
    - **Authorization**: what are you allowed to do?
    - **Confidentiality**: No one may look into this request/response
    - Data Integrity: No one may change this request/response
- **How does OAuth2 works?**
    - Token based authentication and authorization framework
        1. User does request
        2. Get token request is sent to OAuth2 service
        3. OAuth2 service returns token
        4. User calls service with the obtained token
        5. Service verify the authenticity of the token by calling OAuth2 service
        6. OAuth2 service returns the user roles for this token
        7. Service checks if this user is allowed to do such action or not
- **JWT tokens**
    - JWT (JSON Web Tokens) provides a standard structure for OAuth tokens
        - Small
        - Cryptographically Signed
        - Self contained
        - Extensible
- What are the problems with **Traditional Messaging Systems**
    - If the consumer is temporarily unavailable (or very slow) the messages middleware has to store the messages
        - This restricts the volume of messages and the size of the messages
        - Eventually message broker will fail
    - If the consumer has a bug, and handles the messages incorrectly, then the messages are gone
        - Not fault-tolerant
- **What are the characteristics of apache kafka**
    - High throughput
    - Distributed
    - Unlimitedly scalable
    - Fault-tolerant
        - Reliable and Durable
    - Loosely coupled Producers and Consumers
    - Flexible publish-subscribe semantics
- **What's apache zookeeper?**
    - Maintains metadata about a cluster of distributed nodes
        - Configuration information
        - Heath status
        - Group Membership
- **What's kafka partition?**

    - Each topic has one or more partitions
        - This is configurable
    - Each partition is mantained by one or more brokers.

Lesson 13

- **What is blackboard used for?**
    - Used for non deterministic problems
        - There is no fixed straight line solution to the problem
    - Every component adds its information to the blackboard.
- **What's blackboard?**
    - Common data structure
        - Extension is no problem
        - Change is difficult
    - Easy to add new components
    - Tight coupling for data structures
    - Loose coupling for:
        - Location
        - Time
        - Technology
    - Synchronization issues
- **Advantages and Disadvantages of Blackboard**
    - Advantages
        - Easy to add new components
        - Components are independent of each other
        - Components can work in parallel
    - Disadvantages
        - Data structure is hard to change
            - All components share the same data structure
        - Synchronization issues
- **Event Sourcing**
    - Instead of storing the state of an entity in a database, you store a series of events that lead up to the state
    - Storing all the events increases the analytical capabilities of a business
- **Advantages of Event Sourcing**
    - You don't miss a thing
        - Business can analyze history of events
        - Bugs can be solved easier
    - Can be replayed
    - Events are immutable
- **What's a snapshot?**
    - Intermediate steps in an event stream that represent the state after replaying all previous events
        - Can increase performance when streams are very long
- **What is batch processing?**
    - First store data in the database
    - Then do queries (map-reduce) on the data

    - Queries over all or most of the data in the dataset
    - Latencies in minutes to hours
- **What's stream processing?**
    - Handle the data when it arrives
    - Handle event (small data) by event
    - Latencies in seconds or milliseconds
    - Works good for applications with
        - high volume data
        - high frequency changes
- **What's stream?**
    - Stream is a sequence of events implemented with distributed messaging system