**System Design : Final:**

System design points:

https://www.youtube.com/watch?v=bUHFg8CZFws&list=PL0evysyBWYO4n2wkp1UgqpUfSsXnu5EMf&index=7

**===========STEP: 1=========**

Func:

System Behaviour,APIs

Non-Func:

Fast; performace; Fault tolerance; CAP Theroem(Avail. Vs Consietency)

(UPSC)

User

Marketing / Indiviudla Users

Scale

Read or Write heavy

Rate Limiting

Any Spikes Excpected

Performance

Read to Write Dleay Expected??

Streaming Vs Batch

Cost

if Devep>maintenece

Opensource

else

public cloud domain installation

=========== STEP 2=============

High Level Components

2.a) Data Model;

Persistemce Approach: Stream Vs Batch

Pros and Cons for:

Streaming(Event Based/ Individual Click):

Fast Writes; Can slice data as needed;Re-calculation can be done

Slow Reads and Costly for huge data

Batch(Aggregate Data)

Fast Read; Ready for decision Making

Need data Aggregation Pipeline and Can query only the way Data is agregatted

2.b) Where to persist: NOSQL vs SQl

Scalability vs Availability

ProxyServer??????????

Database Sharding? How it works????????

NoSql: Uses Gossip protocol in cassandra: Quorum read/write

=========== STEP 3===================

dATA pROCESSING:

3.a) Fast ---> In Memory

Scalable--> Partition

Should data be pre-aggregated in processing service: means to increment count to database or

Increament in memory and calculate final count and update database in after few minutes?

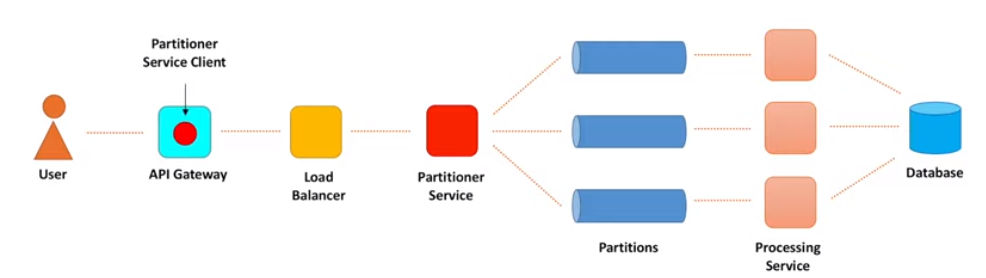
Push Or Pull ProcessService???

Push: the Other events push the data

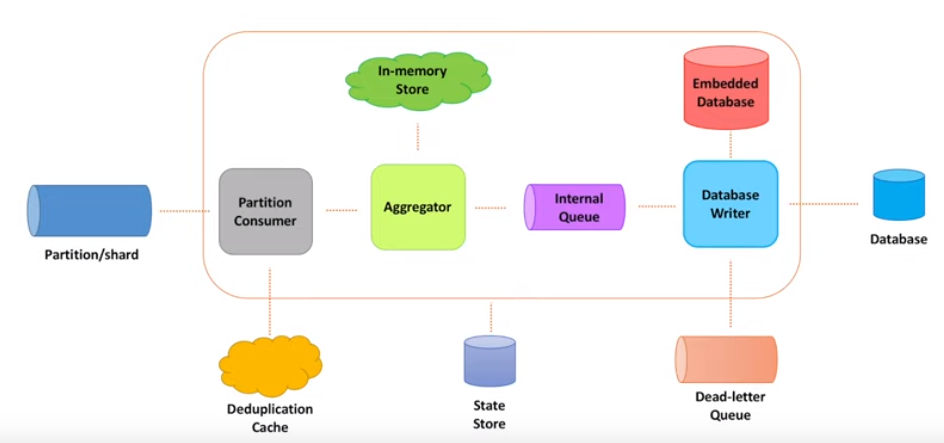
Pull where in the data will be first saved in temporary storage as Queue/kafka and then pushed to memory counter that updates into the database

Checkpointing Vs partioning???????????????????

Ingestion Path Components:

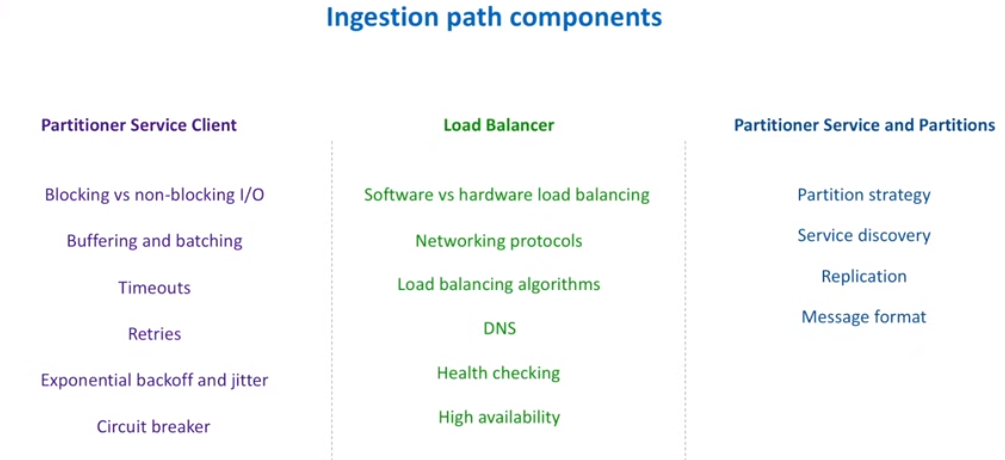


Data Ingestion:



**Blocking Vs Non-Blocking IO System???**

In non-blocking IO the thread progress can be tracked using thread Trace



**SOLID:**

S: Single respo

O: Open for extension closed

L: Linksov Substitution

I: Interface degregation

D: Dependency Inversion

Same as dependency injecton in spring framework

Need for API Gateway:

**API security: Role based; Load Balancing; Circuit Breaking?????**

API Gateway: Single point of entry

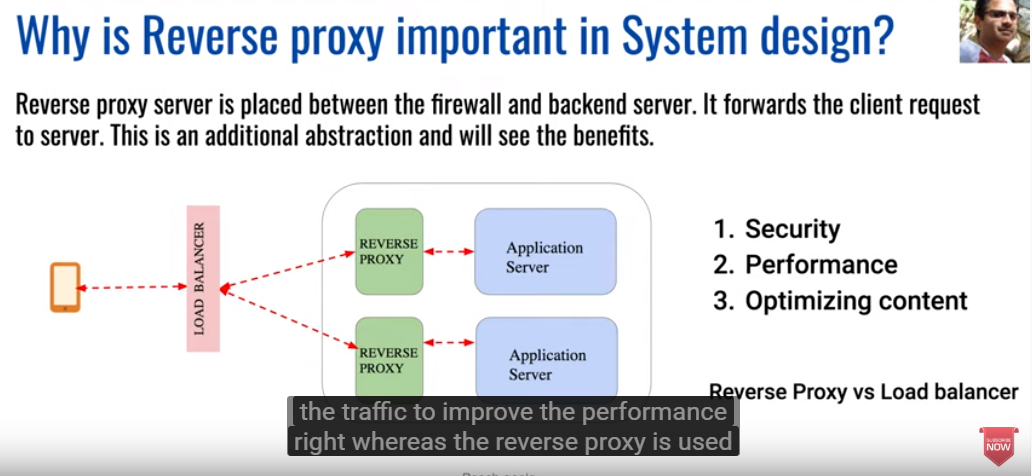
**Api Gateway Vs Proxy??**

Reverse proxy placed between load balancer and BackEnd server

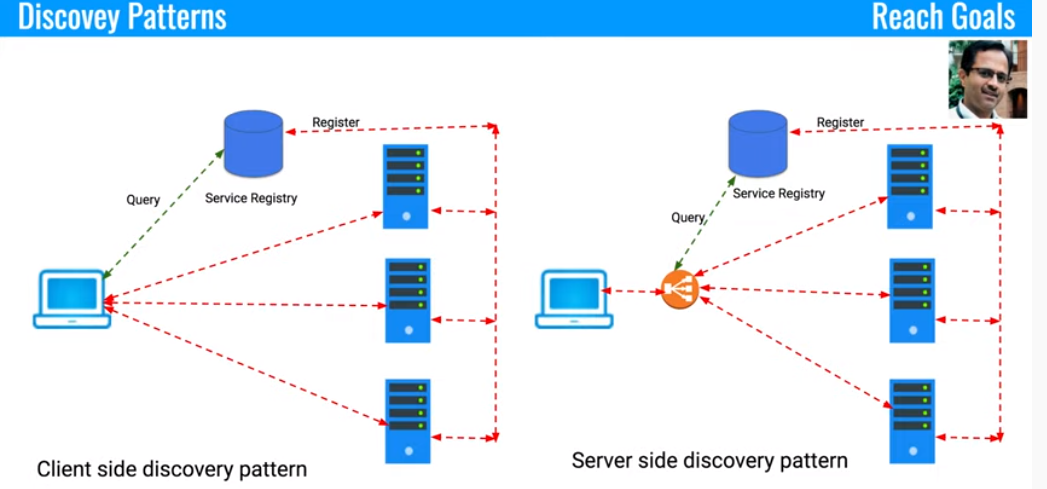
**Reverse Proxy Vs Load balancer:**

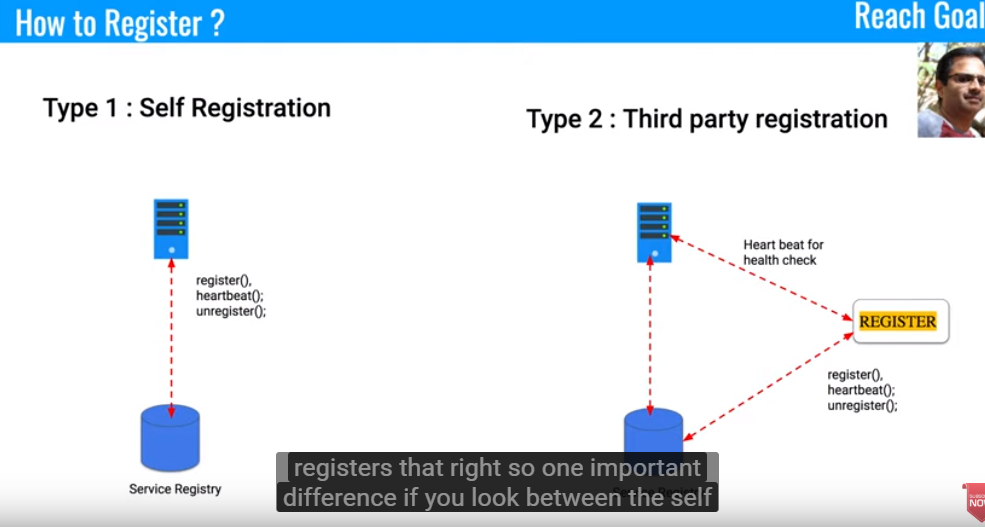
* + Hides the backend infrastructure
  + Used to cache
  + Reverse proxy can work as load balancer also.
* Caching web acceleration
* Canary Deployment
* Security against external traffic
* Single entry point
* Can work as load balancer

**Microservices Registry:**

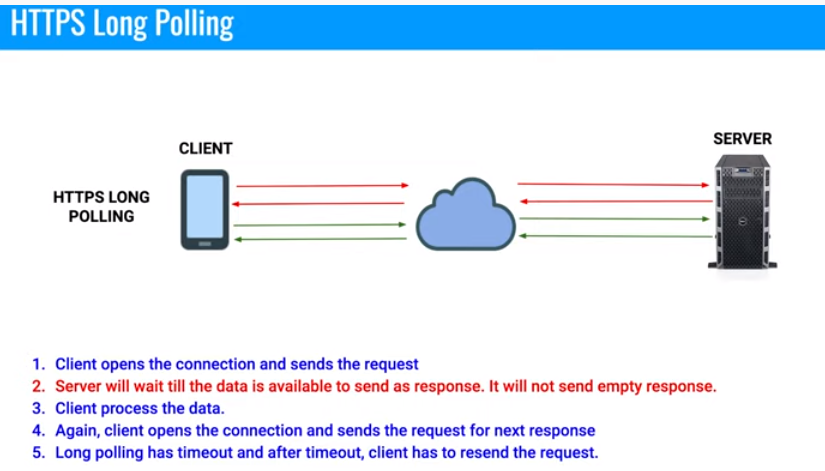


MicroServices Discovery Pattern:





Http Polling:?????



**Web Sockets:**

**WS://**

Bidirectional

**MicroServices Pattern**

**REST API??**

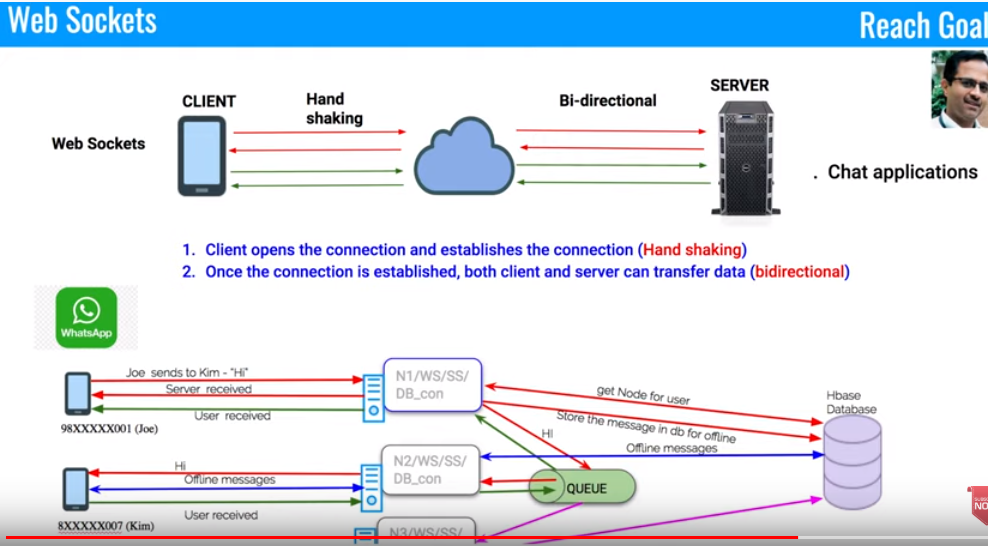
**Security Patterns???**

<https://www.youtube.com/watch?v=GMlHsF1JmsE>

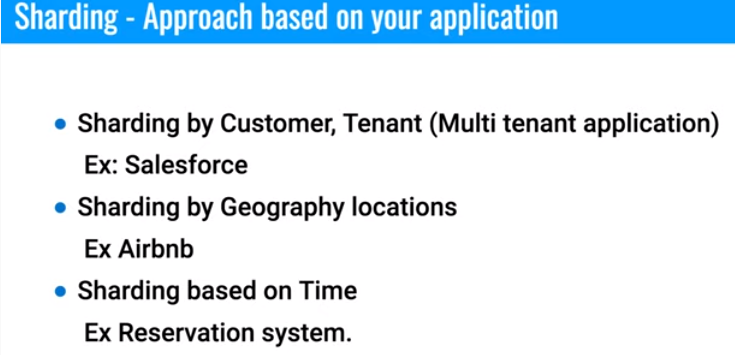
**AJAX polling:**

Request based on javascript configuration

HTTP Polling doesn’t send the empty response but wait for server to respond



**DataBase Sharding:**



**Data Model:**

**Payment gateway??**

**Schema/ Data Calculations???**

**DataCapacity Model**

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>..

**STEP4:**

**WHATSaAPP:**

**WhatsApp:**

**1. Authentication**

**2. one to one ; user online**

**3. Group Messaging**

**4.MultiMedia Message**

**use**

**Allocate the WhatsApp number to a Node; called pre-allocation/Provisiong the Node**

**Payment gateway:**

1. **Payment QR Code:**

**URl in 2d image**

Qr Code:

2d code containing the URl and info

contactless

buyer and seller Schema

WhatsApp:

websocket/https polling

schema:??

Messgae sequencing: Need timestamp for all messages

Pagination for long messages?? how it displays in messages

DB Optimise using the Cache Server(??)

Schema:

User table

Offline\_message\_table

group\_info\_meggase\_table(for group messaging)/ owner\_of\_group

Twitter:

usecase:

Home ScreenTimeSeries

User ScreenTimeSeries

Search

follow

FanOut

SocialGraph / DS

Redis cache farm

load balancer

schema:

Tiny Url: domain\_name/uniqueCode

base 64

Mapping between Actual URL and Tiny URL

API Security CheckList: